Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Theory of Robotics Systems

Planning – getting from A to B

Co-funded by the
Erasmus+ Programme
of the European Union

Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Faculty of Technical Sciences

Ss. Cyril and Methodius University
Faculty of Electrical Engineering and Information Technologies

Zagreb University of Applied Sciences

School of Electrical Engineering
University of Belgrade

Faculty of Physics
Warsaw University of Technology

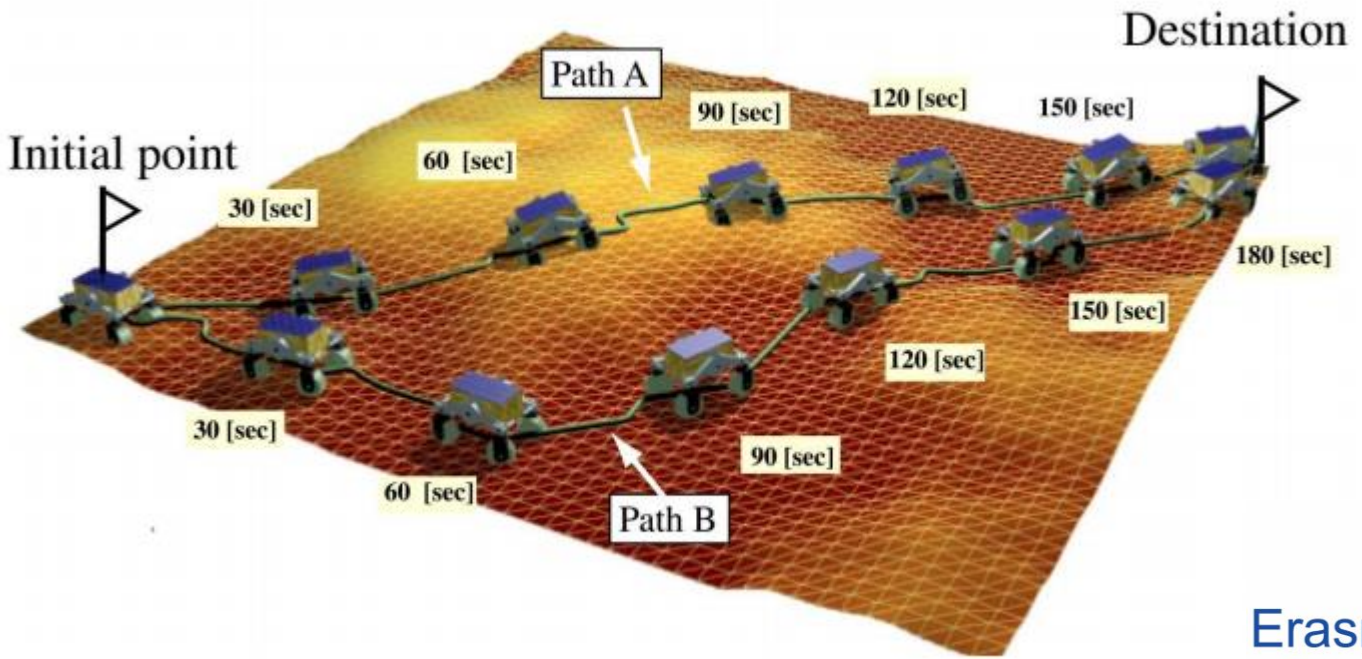
Co-funded by the Erasmus+ Programme of the European Union

Šta znači planiranje?

- Generalno, naš fokus će biti na **planiranju kretanja** u robotici. Određivaćemo set akcija koje će robota odvesti iz poznate pozicije u neku drugu takođe poznatu poziciju.
- Takođe posmatraćemo koja su to **ograničenja** koja mogu uticati na planiranje i od čega potiču kako bismo znali da li je naše planiranje kretanja **izvodljivo**.

Co-funded by the Erasmus+ Programme of the European Union

Šta je planiranje u mobilnoj robotici?



Co-funded by the Erasmus+ Programme of the European Union

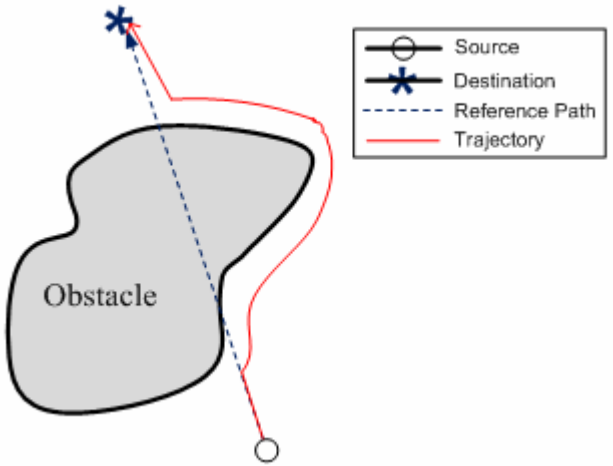
Povezane teme

- Upravljanje:
  - Generalni zadatak dohvatanja (dolaženja) do željenog stanja.
  - Najčešće koristimo **feedback** upravljanje.
  - Uspešnost upravljanja merimo pomoću stabilnosti, robusnosti i otpornosti na poremećaj.
- Planiranje u Veštačkoj Inteligenciji:
  - Bavi se diskretnim problemima.
  - Klasični algoritmi u VI (pretraga grafova, problem trgovačkog putnika, orijentiring) se često primenjuju kao algoritmi za planiranje u robotici.

Co-funded by the Erasmus+ Programme of the European Union

Istorija planiranja kretanja u robotici

- Klasična robotika (sredina 70-ih)
  - Tačan model, bez senzora
- Reaktivno planiranje trajektorije (sredina 80-ih)
  - Bez modela, oslanja se isključivo na senzoriku

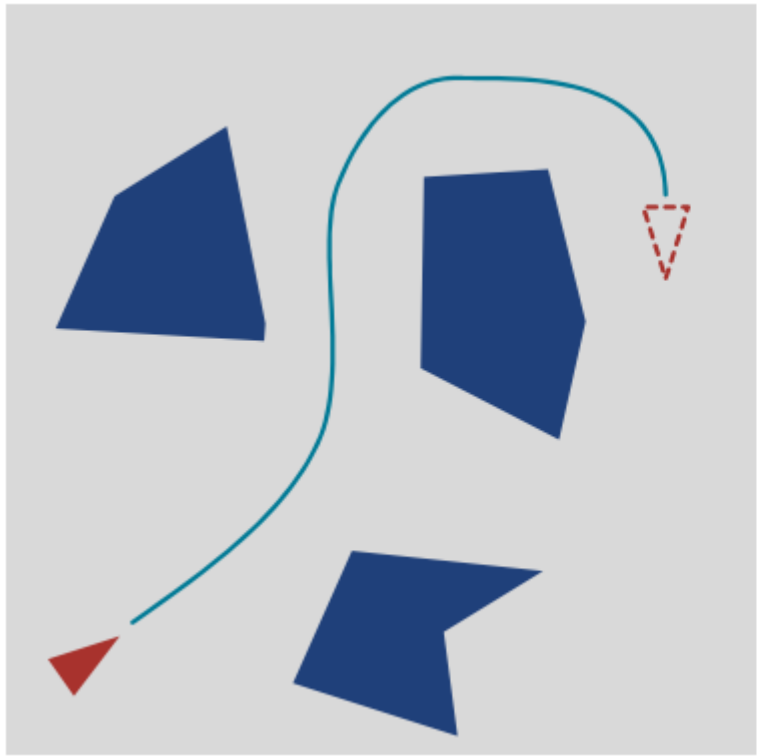


Co-funded by the Erasmus+ Programme of the European Union

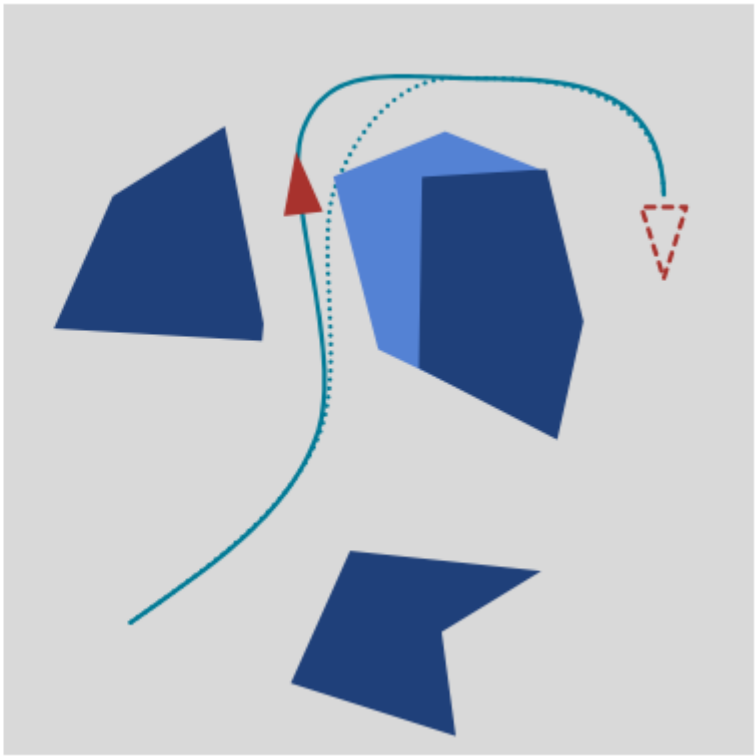
Istorija planiranja kretanja u robotici

- Hibridno / hijerarhijsko planiranje (početak 90-ih)
  - Model za globalno planiranje, viši nivo
  - Reaktivno (zaobilaženje prepreka), niži nivo
- Probabalističko planiranje (sredina 90-ih)
  - Uključuje nesigurnost modela i senzora u sve nivoi planiranja

Co-funded by the Erasmus+ Programme of the European Union



Planirano



Izvršeno

Co-funded by the
Erasmus+ Programme
of the European Union

Uloga navigacije

- Ukratko, naći način kako robot može da se pomeri sa jednog mesta na drugo, poštujući sva ograničenja.
- Neke od pretpostavki:
  - Dovoljno dobro možemo da **predstavimo** okolinu i robota
  - Znamo gde se robot **nalazi** i gde **treba da dođe**
  - Posedujemo **model kretanja** robota
- Obično se predstavlja kao problem optimizacije – minimizacija cene (vreme, udaljenost, energija), poštujući ograničenja.

Co-funded by the Erasmus+ Programme of the European Union

Planiranje

- Robot se nalazi na startnoj poziciji S, treba da dođe do tačke G:
  - Potrebno je pronaći put koji vodi robota od S do G.
- Izazovi:
  - Prepreke?
  - Dinamika?
  - Izvodljivost?
- Optimizacion problem !

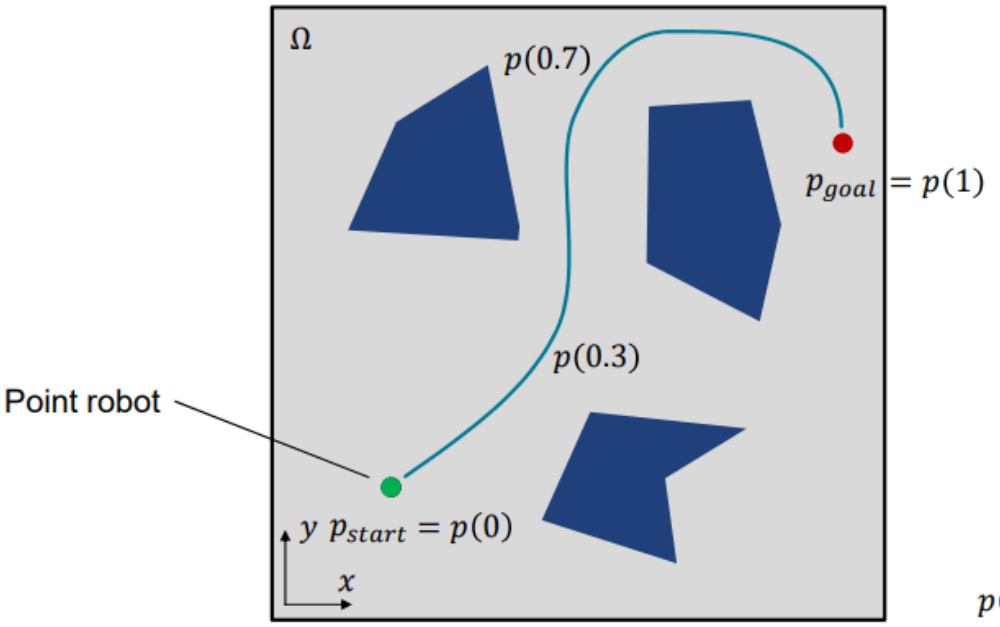
Co-funded by the Erasmus+ Programme of the European Union

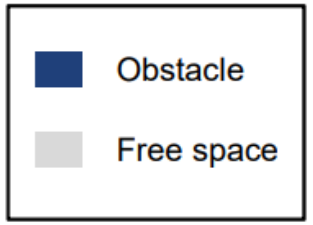
Reprezentacija

- Veoma je bitno na koji način robot predstavlja i razume svet.
- Uglavnom se moraju primeniti određene metode pojednostavljivanja.
- Odabirom odgovarajuće reprezentacije sveta, određeni algoritmi se mogu primeniti kako bi rešili problem planiranja.

Co-funded by the Erasmus+ Programme of the European Union

Reprezentacija – radni prostor i putanja





$$\omega_{free} = \Omega - \omega_{obs}$$

$$p: [0,1] \rightarrow \omega_{free}$$

$$p(t): \forall t \in [0,1], \; p(t) \in \omega_{free}$$

$$p(0) = p_{start}$$

$$p(1) = p_{goal}$$

Co-funded by the Erasmus+ Programme of the European Union

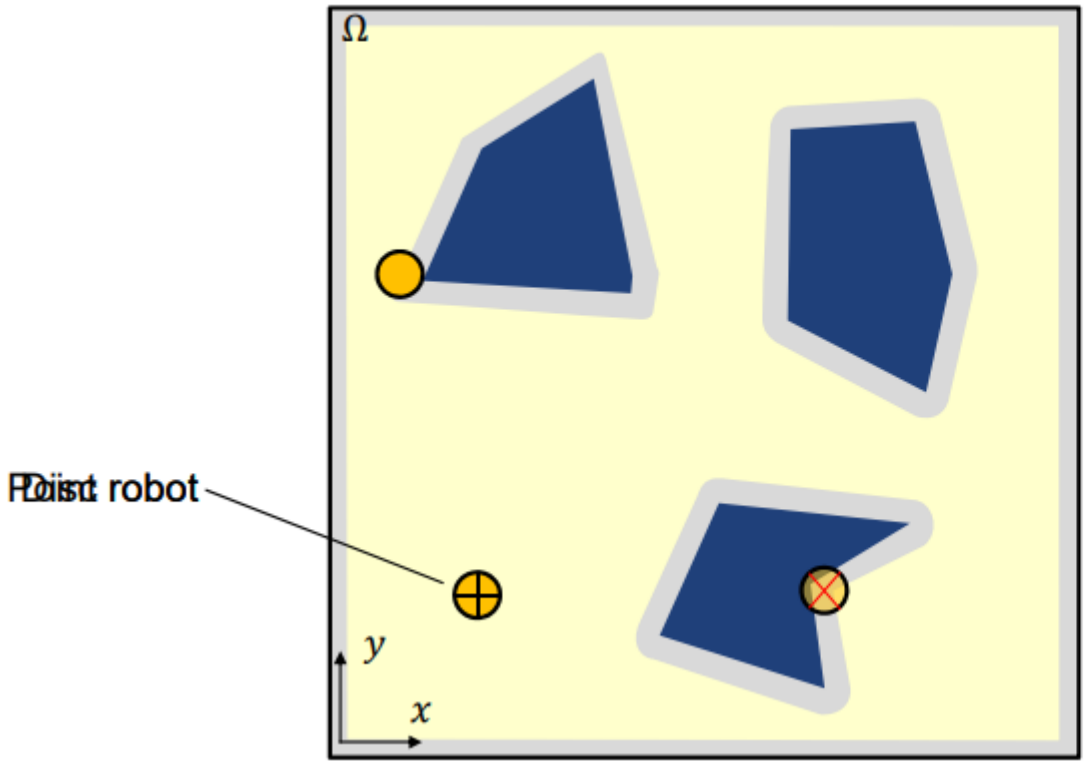
Reprezentacija – radni prostor i putanja

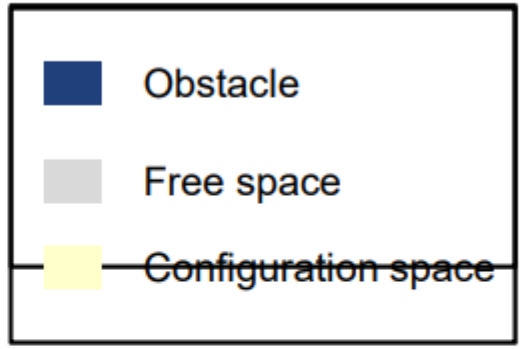
- Radni prostor uglavnom predstavlja sam svet (mapu), po mogućstvu nezavisno od robota. Uglavnom na određeni način predstavlja koje su oblasti slobodne a koje ne.
- Konfiguracijski prostor opisuje sva stanja u kojima se robot u datom prostoru može naći.
- Od sad smatramo da naš robot nije tačka u prostoru već sam zauzima određeni prostor.

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

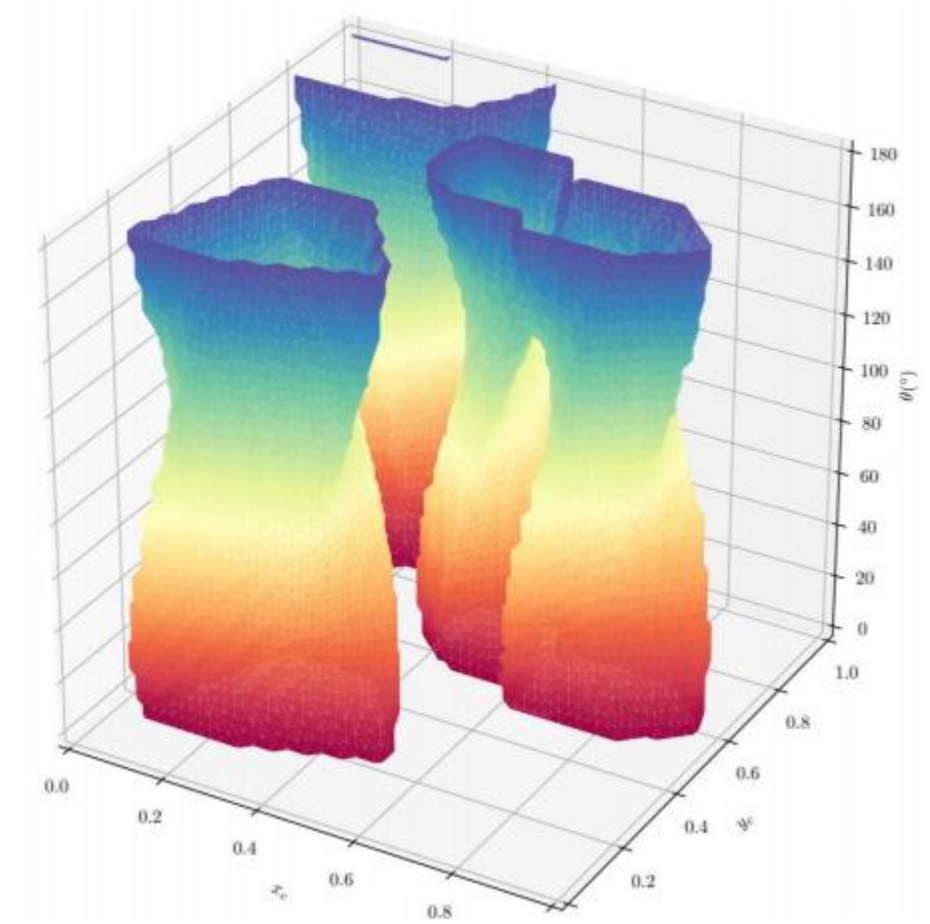
Konfiguracijski prostor





Co-funded by the
Erasmus+ Programme
of the European Union

Konfiguracijski prostor



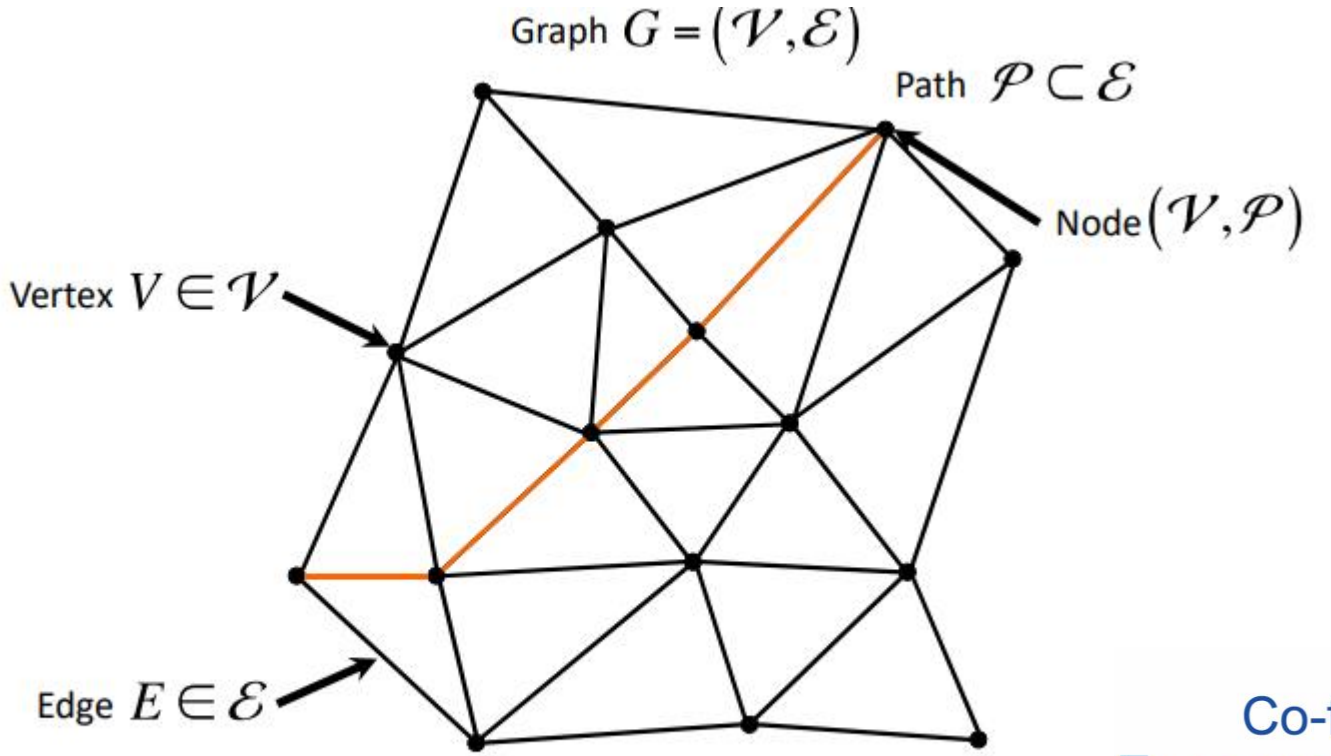
Co-funded by the
Erasmus+ Programme
of the European Union

Kontinualna vs Diskretna reprezentacija

- Iako za kontinualnu reprezentaciju posedujemo dobar matematički alat, uglavnom nam takva predstava nije zgodna.
- Kako računari čuvaju podatke digitalno, potrebno je naći pametan i efikasan način za kreiranje (aproksimacije) diskretne reprezentacije.
- Veoma je uobičajeno da problem planiranja predstavimo kao problem pretrage grafa, jer za pretragu grafova postoji dosta već razrađenih algoritama.

Co-funded by the Erasmus+ Programme of the European Union

Terminlogija

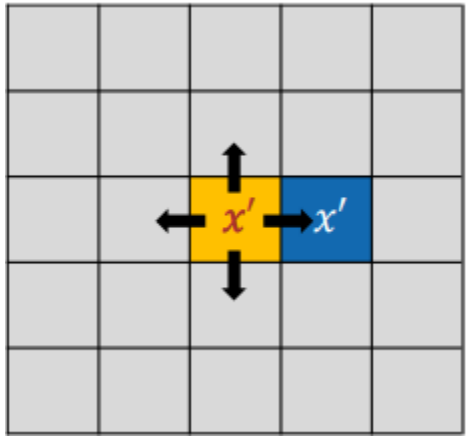


Directed graph: edges have direction
Weighted graph: edges have costs

Co-funded by the Erasmus+ Programme of the European Union

Diskretna reprezentacija prostora stanja

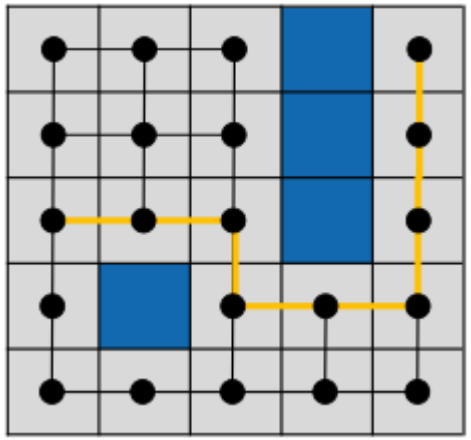
- Redukujemo kontinualna stanja na skup konačnih diskretnih stanja.
  - $x \in X$
- Definišemo izvodljiv skup akcija za svako stanje
  - $A(x) = \{a_0, a_1, \ldots, a_n\}$
- I svakoj od akcija pridružujemo funkciju tranzicije
  - $f(x, a) = x'$



Co-funded by the Erasmus+ Programme of the European Union

Grid → Graf

- Primeniti:
  - Stanja se uzimaju kao vrhovi
  - Prelazi kao usmerene ivice
- Rezultat je graf
- Dodajemo:
  - Početni nod, x_s
  - Završni nod, x_g
  - Funkcija cene $C: X \times A \rightarrow \mathbb{R}^+$
- Traženje najkraćeg puta možemo tretirati kao problem pretrage grafa.



Co-funded by the Erasmus+ Programme of the European Union

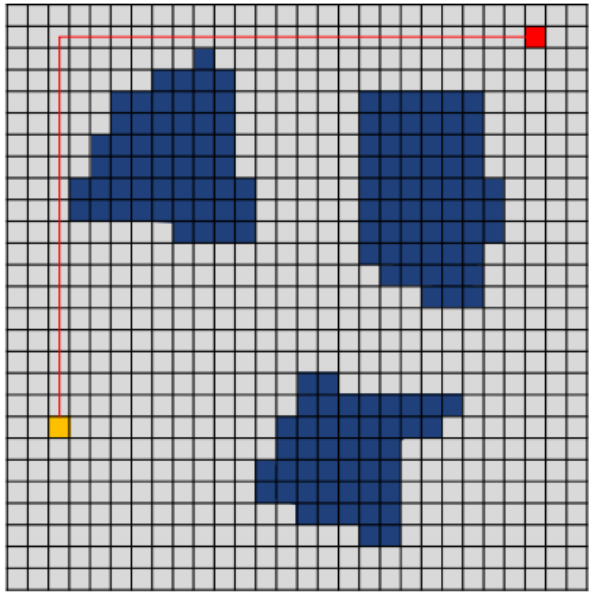
Problemi sa reprezentacijom pomoću grid-a

- Uglavnom se neki sitni detalji i preciznost mogu izgubiti.
- Odabir prave rezolucije može

da bude veoma izazovan zadatak

(mapiranje multi-rezolucijom)

- Izlazna putanja može biti ograničena.
- Loša rezolucija za mape velikih dimenzija.

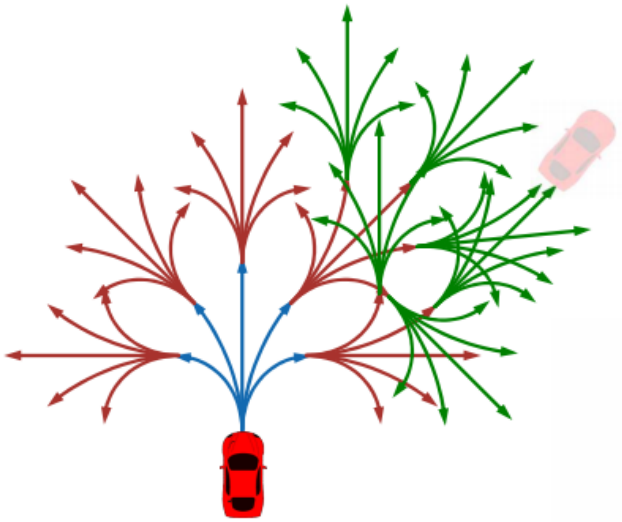


Co-funded by the Erasmus+ Programme of the European Union

Grid rešetka

- Kreira set mogućih primitivnih kretanja,

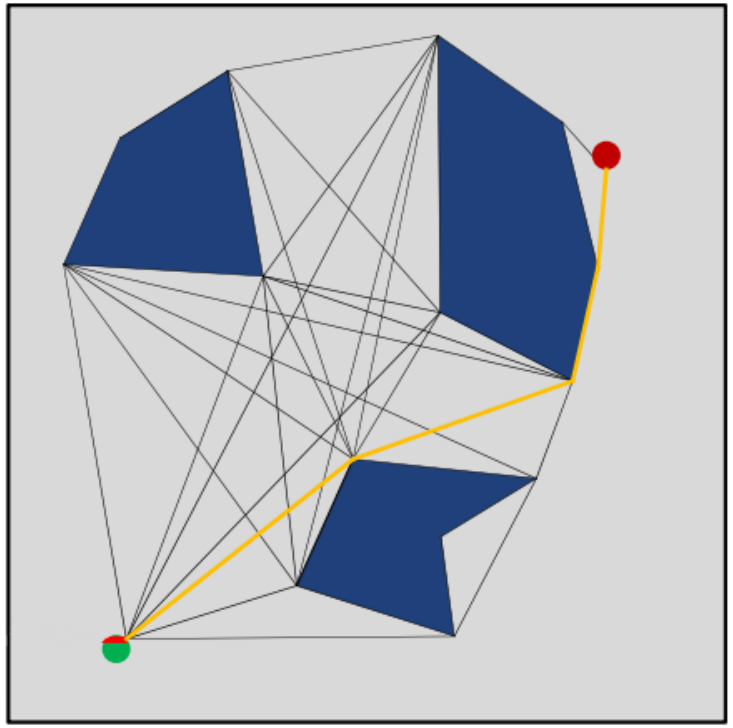
i konstruiše drvo (graf) koje sadrži određena kretanja u sekvenci (putanja)



Co-funded by the Erasmus+ Programme of the European Union

Grafikon vidljivosti

- Kreira ivice grafa između svih parova vidljivih čvorova.
- Pretraga se vrši prema dobijenom grafu.
- Optimalan put je putanja kretanja.
- Ograničeno kretanje na prave putanje.



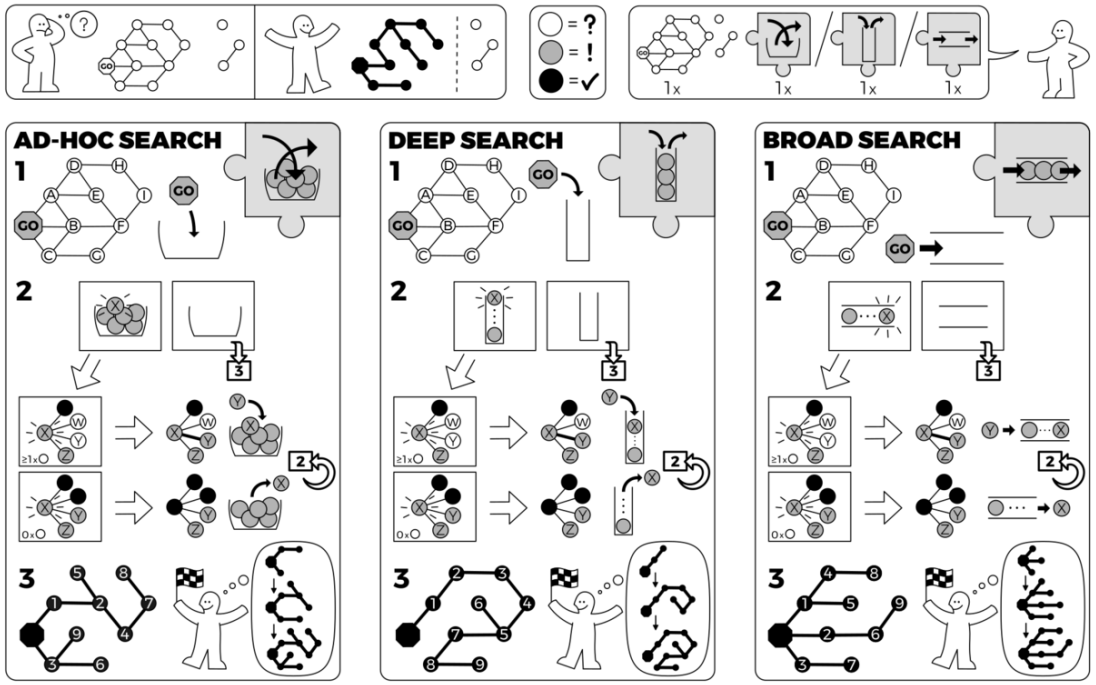
Co-funded by the Erasmus+ Programme of the European Union

Planiranje kretanja

- Planiranje kretanja: deljenje zadatka kretanja na diskretna kretanja:
  - Robot mora da stigne od početne lokacija na cilj.
  - Robot mora da zaobiđe zidove, da uspešno izbegne stepenice, …
  - Idealno, putanja mora da bude najefikasnija (najkraća)
- Ulazi algoritma:
  - Opis zadatka (početna i krajnja pozicija)
  - Ograničenja (fizička ograničenja robota, prepreke, …)
  - Nesigurnosti robota i okruženja
- Izlaz algoritma:
  - Putanja robota
    - Set tačaka koje treba da poseti kako bi otišao od tačke A do tačke B

Co-funded by the Erasmus+ Programme of the European Union

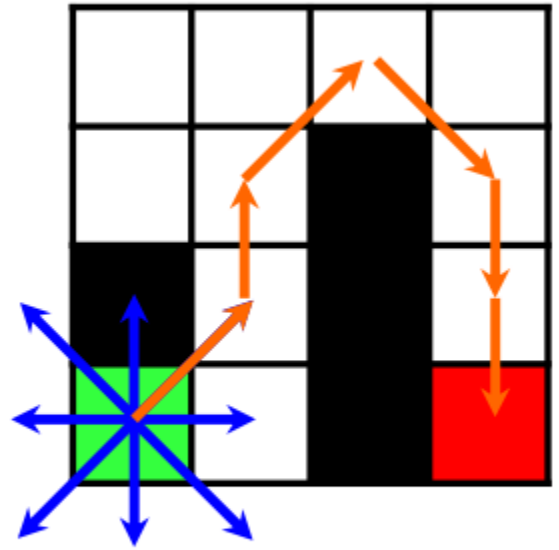
Metode pretrage grafova



Co-funded by the
Erasmus+ Programme
of the European Union

Primer 1

- Putanja od zelene ćelije do crvene
- 8 koraka iz jedne ćelije
- Koja je najkraća putanja?
- Šta ako imamo 4 moguća koraka iz ćelije.

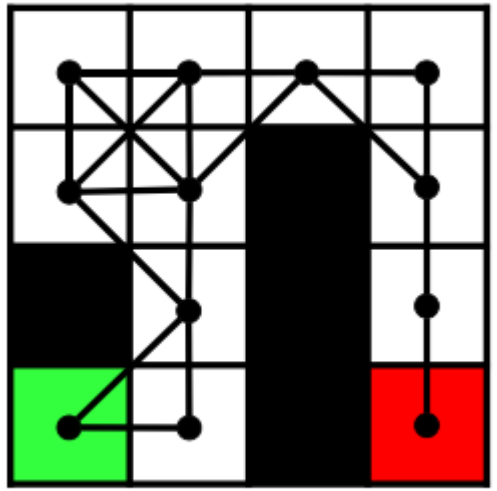


Assumptions about robot motion

Co-funded by the Erasmus+ Programme of the European Union

Primer 1

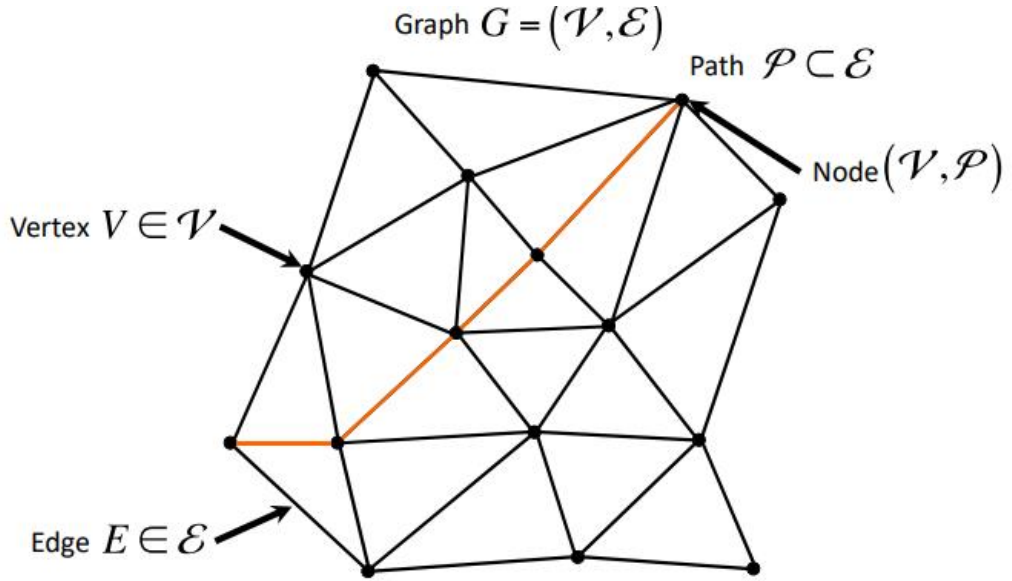
- Pretražimo dobijeni graf.
- Pretpostavljamo da sve ivice grafa mogu kreirati putanju.
- Rešimo problem tako što nađemo putanju od A do B.



Co-funded by the Erasmus+ Programme of the European Union

Pretraga grafova

- Ivice grafa mogu imati pridružene **težine**
  - Cena putanje
- Ima smisla da se razmatraju samo pozitivne težine.
- Algoritmi sa minimalnom cenom nikad ne posećuju isti čvor dva puta.
- Najpoznatije pretrage:
  - Pretraga u širinu (BFS)
  - Pretraga u dubinu (DFS)
  - Dijkstra's Algoritam
  - A*

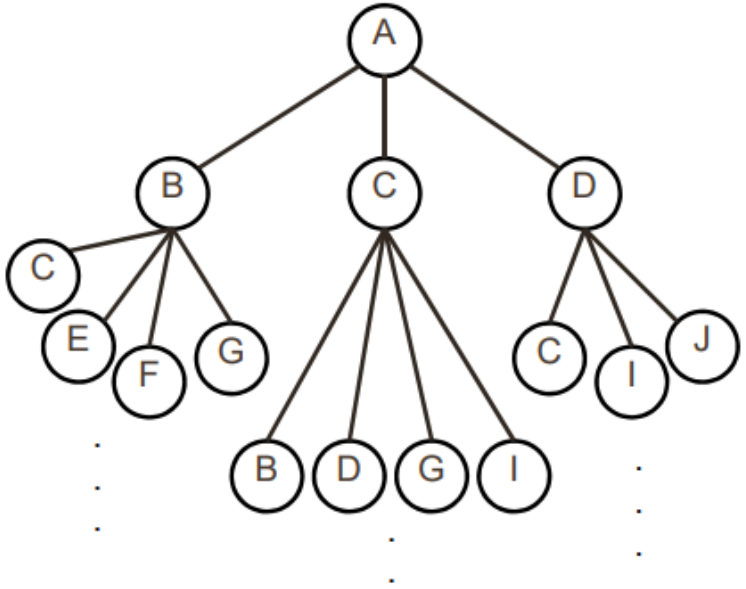


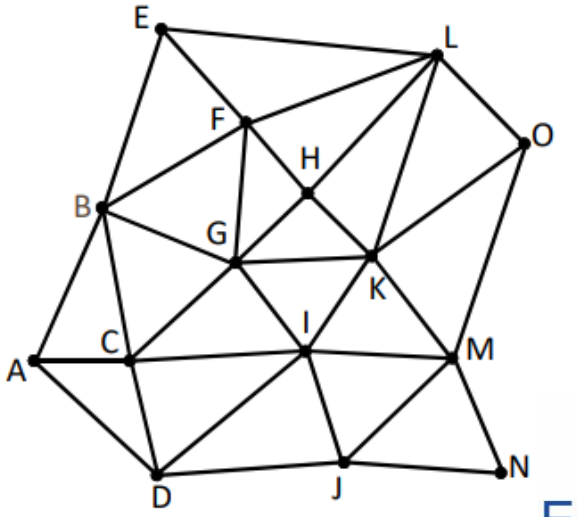
Directed graph: edges have direction
Weighted graph: edges have costs

Co-funded by the Erasmus+ Programme of the European Union

Drvo pretrage

- Konstruišemo „drvo“ pomoću kojeg vršimo pretragu.

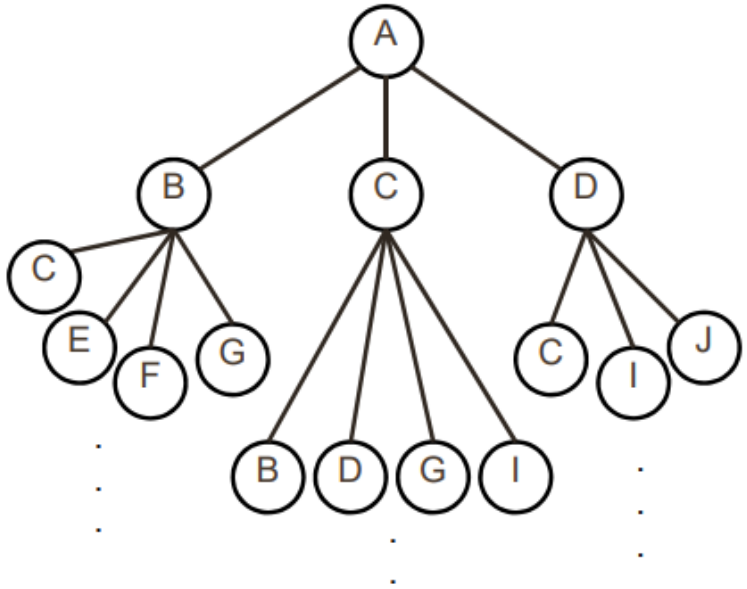




Co-funded by the Erasmus+ Programme of the European Union

Drvo pretrage

- Drvo pretrage:
  - Početno stanje se uzima kao koren
  - Naslednici – skup svih mogućih akcija
  - Uglavnom želimo da izbegnemo konstrukciju kompletnog stabla



Co-funded by the Erasmus+ Programme of the European Union

Osnove pravolinijske pretrage

- Počnemo od korena (početno stanje), širimo drvo sve dok ne dođemo do cilja.
- Proširivanje nodova znači dodavanje njihovih naslednika.
- Treba probati da se nađe rešenje sa što manje proširivanja.
- Otvoren set – čvorovi iz kojih se mogu vršiti proširivanja
- Zatvoren set – svi oni čvorovi koje je algoritam posetio

Co-funded by the Erasmus+ Programme of the European Union

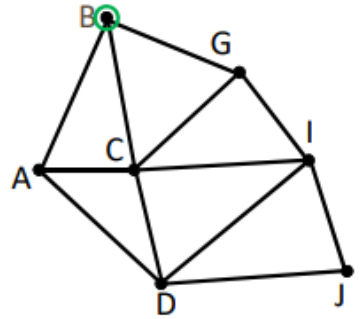
Osnove pravolinijske pretrage

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po širini

```
FORWARD SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```



Open (Q): {B}

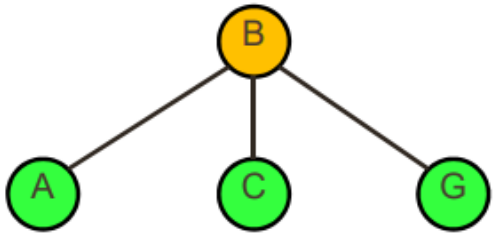
Closed: {}

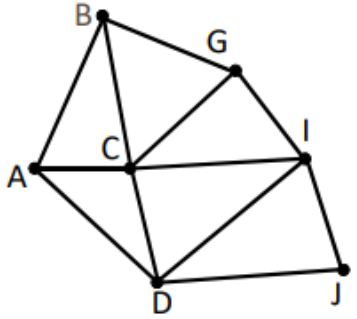
Our (BFS) queue will be FIFO:
- push (*Q.Insert*) onto the end
- pop (*Q.GetFirst*) from the front

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po širini

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```





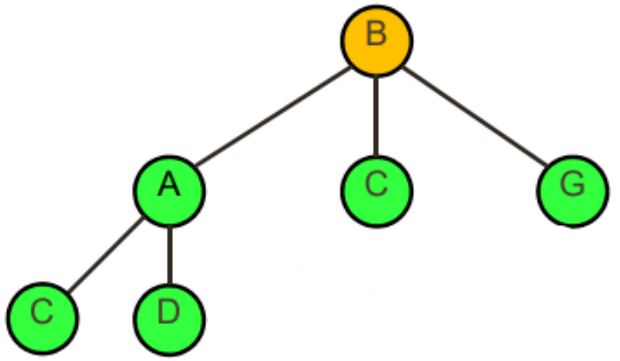
Open (Q): {A,C,G}

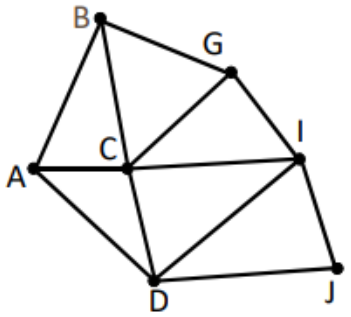
Closed: {B,A,C,G}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po širini

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```



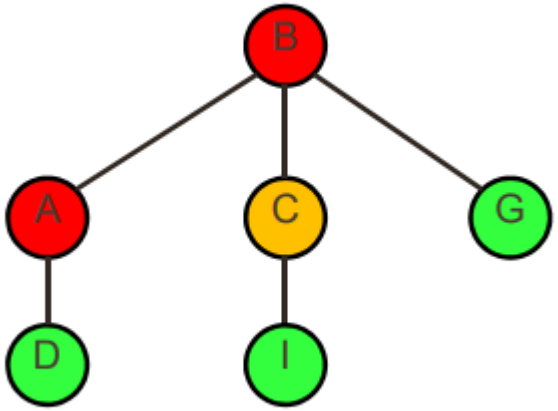


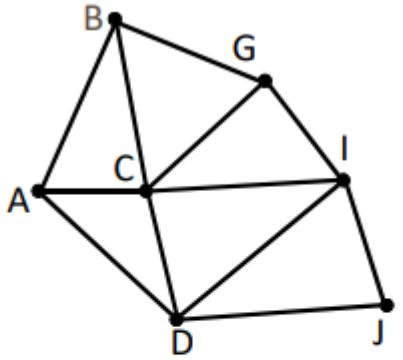
Open (Q): {C,G,D}

Closed: {B,A,C,G,D}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po širini



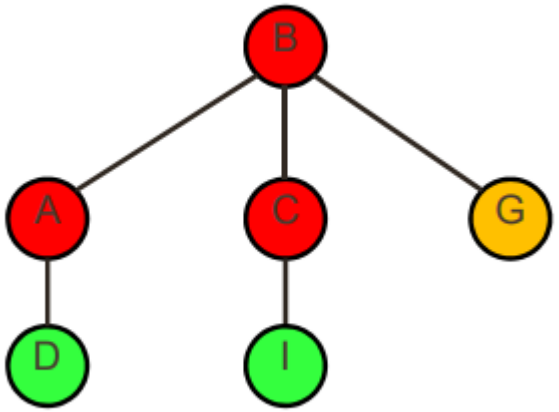


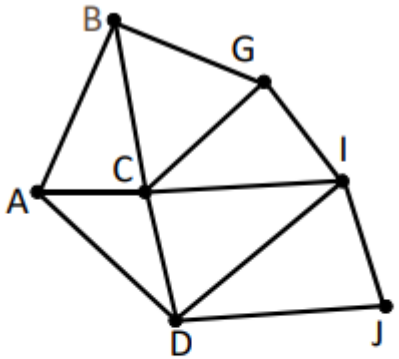
Open (Q): {G,D,I}

Closed: {B,A,C,G,D,I}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po širini





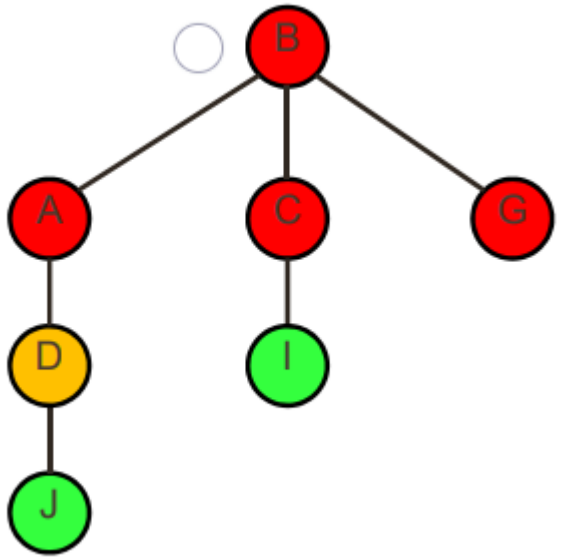
Open (Q): {D,I}

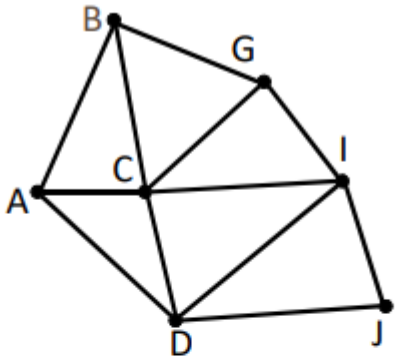
Closed: {B,A,C,G,D,I}

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

Pretraga po širini



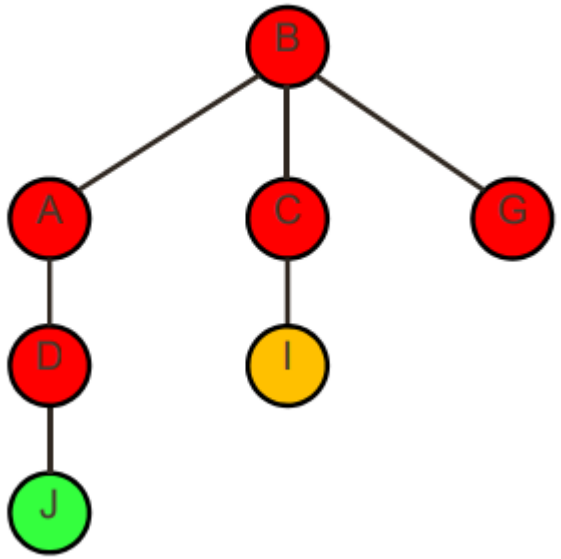


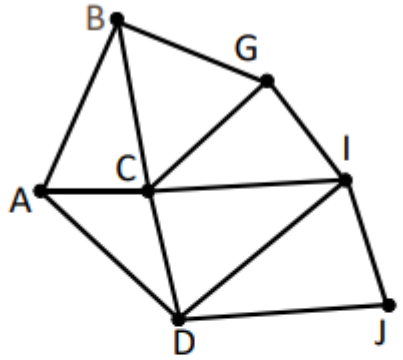
Open (Q): {I,J}

Closed: {B,A,C,G,D,I,J}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po širini



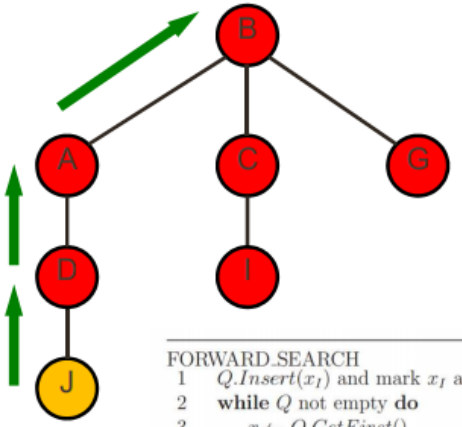


Open (Q):	Closed:
{J}	{B,A,C,G,D,I,J}

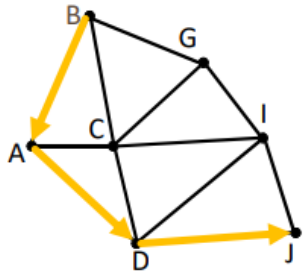
Co-funded by the Erasmus+ Programme of the European Union

Itasdi

Pretraga po širini




```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```



Open (Q): {}

Closed: {B,A,C,G,D,I,J}

Final path solution: B→A→D→J

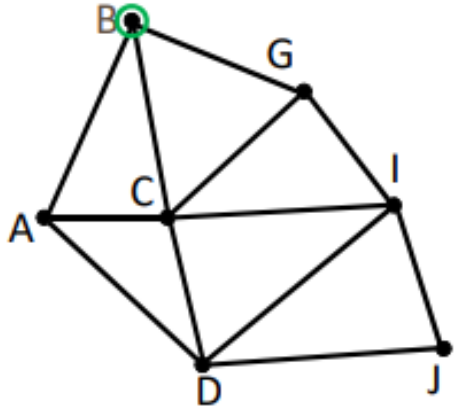
Other solutions may exist but have the same number or more transitions

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

Pretraga po dubini

```
FORWARD SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```



Open (Q): {B}

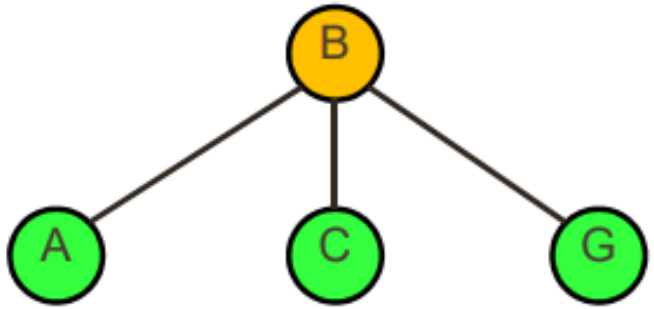
Closed: {}

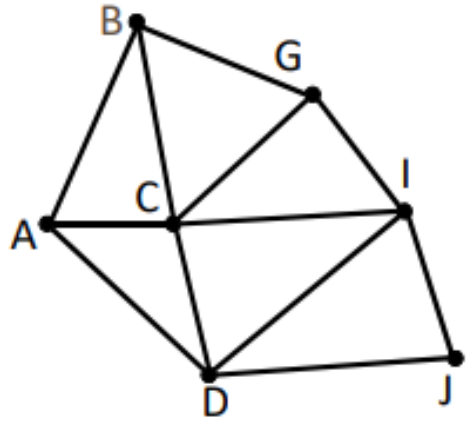
Our (DFS) queue will be LIFO:
- push (*Q.Insert*) onto the front
- pop (*Q.GetFirst*) from the front

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po dubini

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```





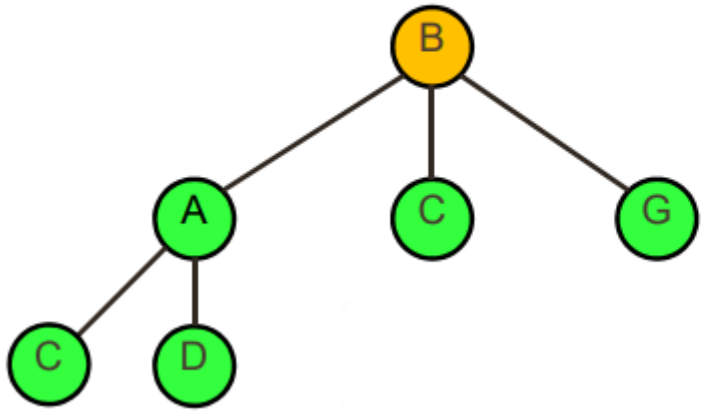
Open (Q): {A,C,G}

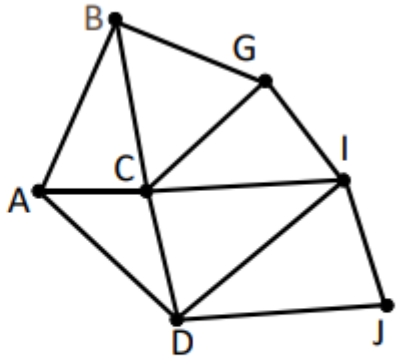
Closed: {B,A,C,G}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po dubini

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```



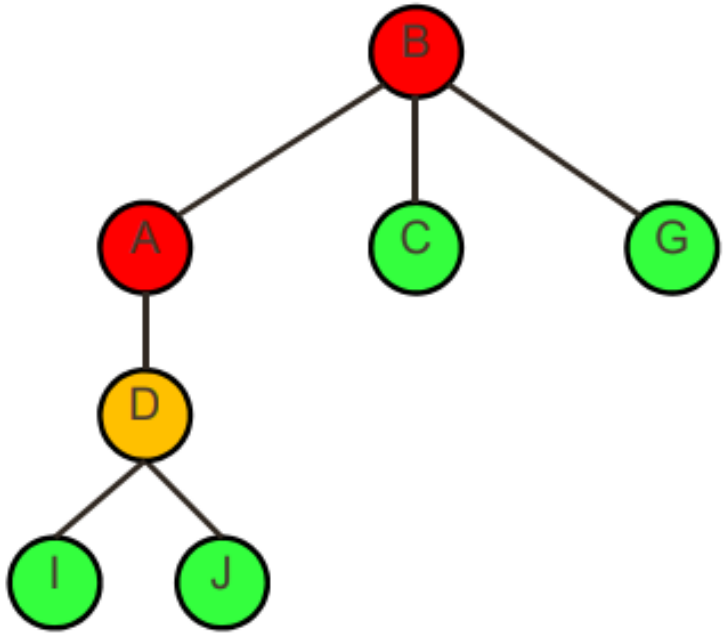


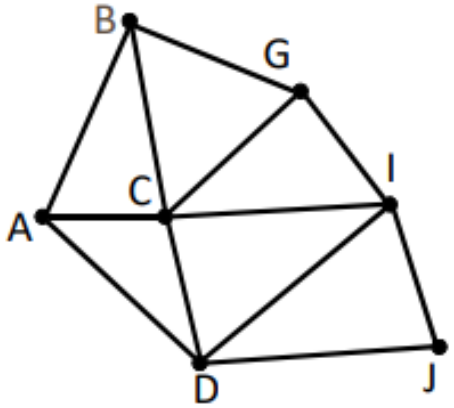
Open (Q): {D,C,G}

Closed: {B,A,C,G,D}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po dubini



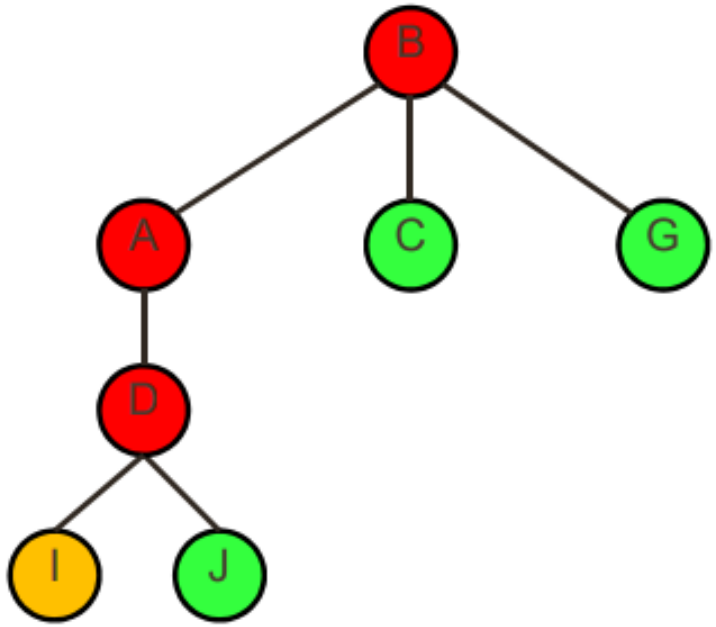


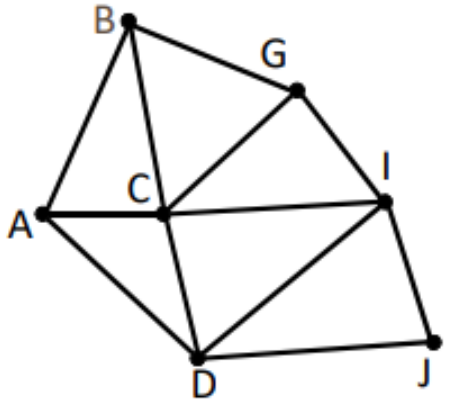
Open (Q): {I,J,C,G}

Closed: {B,A,C,G,D,I,J}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po dubini





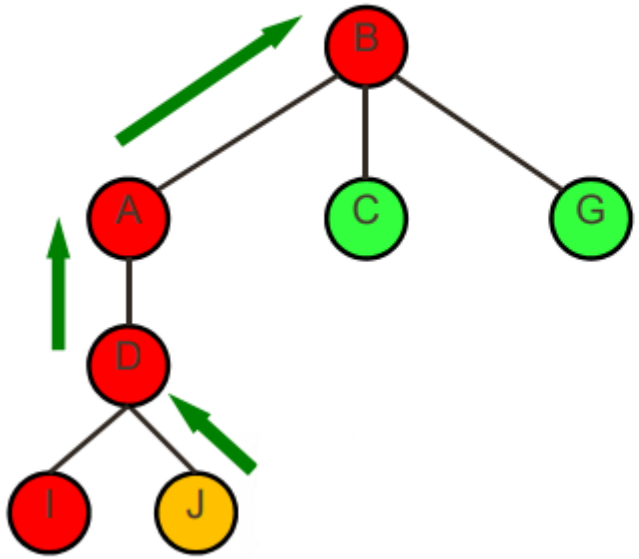
Open (Q): {J,C,G}

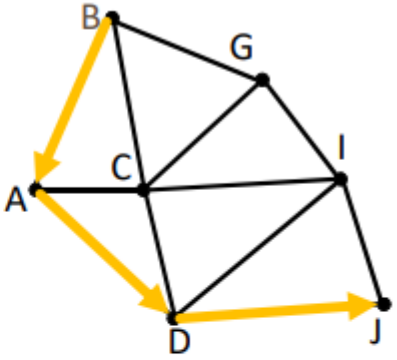
Closed: {B,A,C,G,D,I,J}

Co-funded by the Erasmus+ Programme of the European Union

Pretraga po dubini

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```





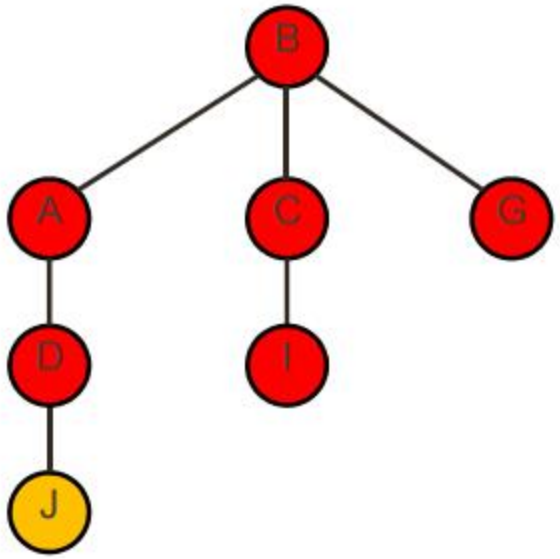
Open (Q): {}

Closed: {B,A,C,G,D,I,J}

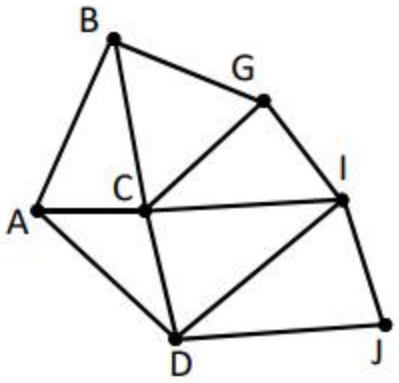
Final path solution: B→A→D→J

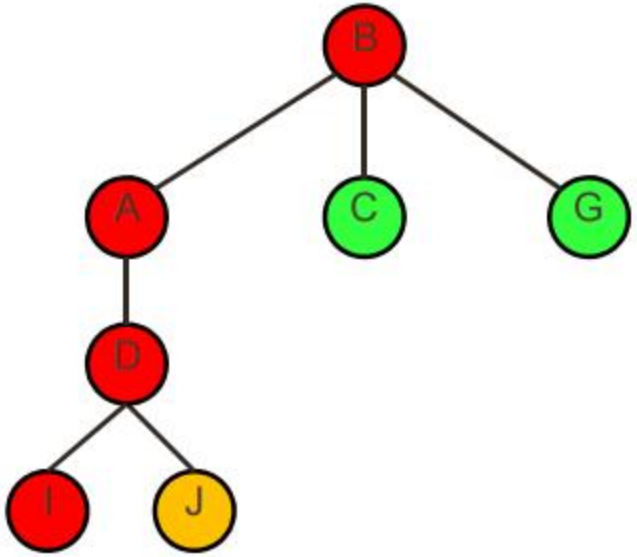
Co-funded by the Erasmus+ Programme of the European Union

Uporedna analiza



BFS



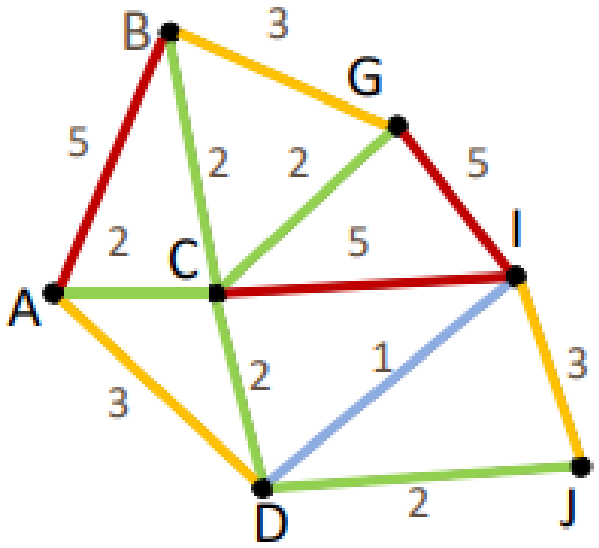


DFS

Co-funded by the
Erasmus+ Programme
of the European Union

Akcije sa cenom

- Kako rešiti slučaj ako određena putanja ima veću cenu nego neka druga?
- Najpopularniji algoritmi Dijkstra's i A*



Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

- Objavljen od strane Edsger Dijsktra 1959.
- Otvara čvorove sa najmanjom cenom najbliže od korena grafa.
- Najkorišćeniji algoritam za rešavanje problema trgovačkog putnika.

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

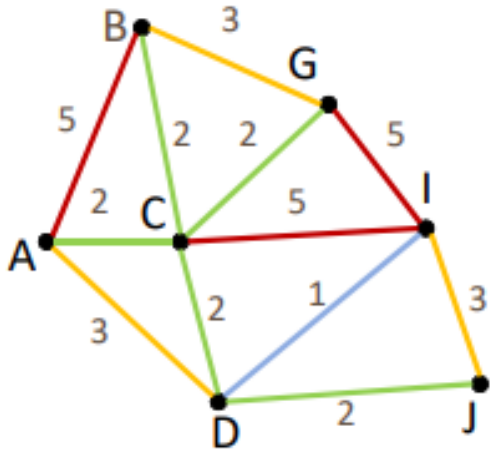
```
FORWARD SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```

B(0)

Open (Q):	Closed:
{B(0)}	{B(0)}

Our Dijkstra queue will be ordered by cost to arrive:

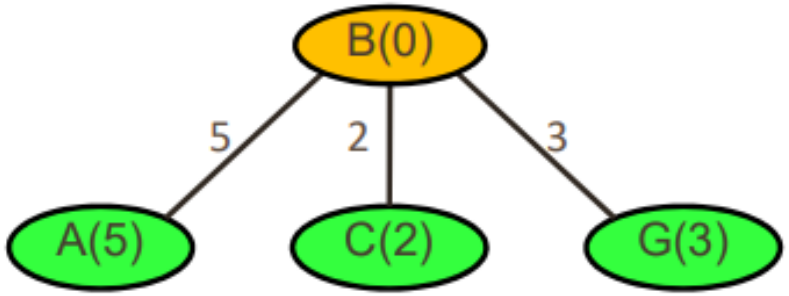
- push (*Q.Insert*) by cost
- pop (*Q.GetFirst*) from the front, and add it to the closed list

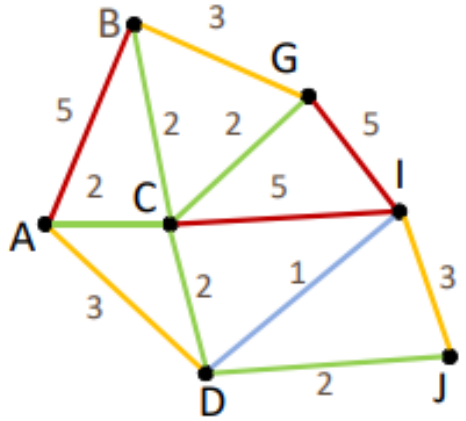


Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```





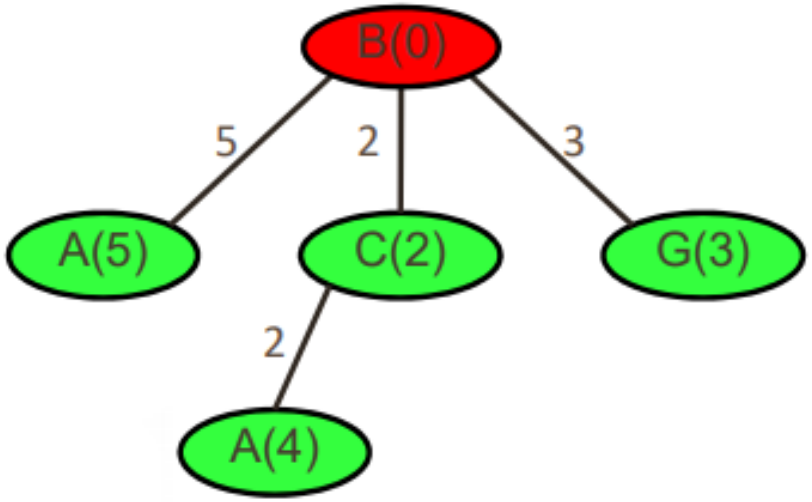
Open (Q):
{ C (2),
G (3),
A (5) }

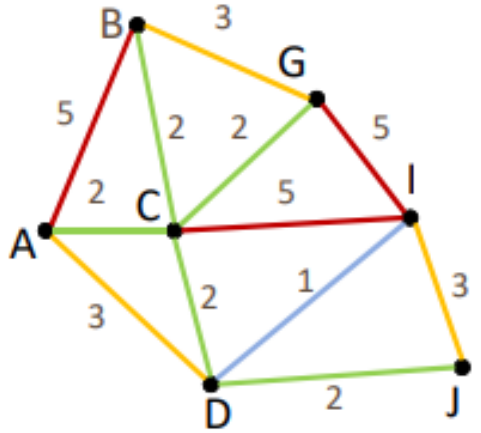
Closed:
{ B (0) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```





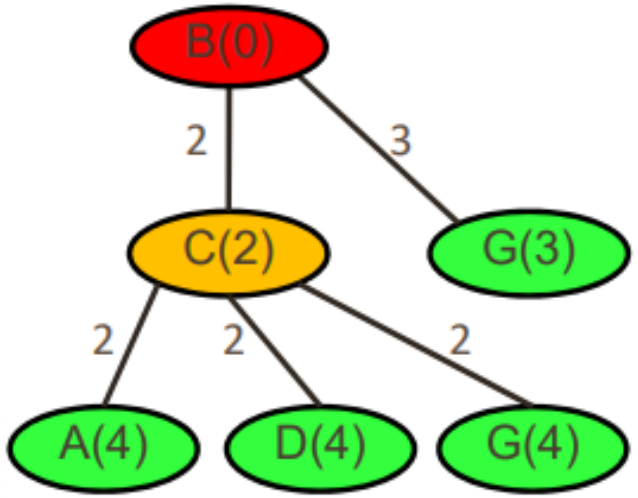
Open (Q):
{ G (3),
A (4) }

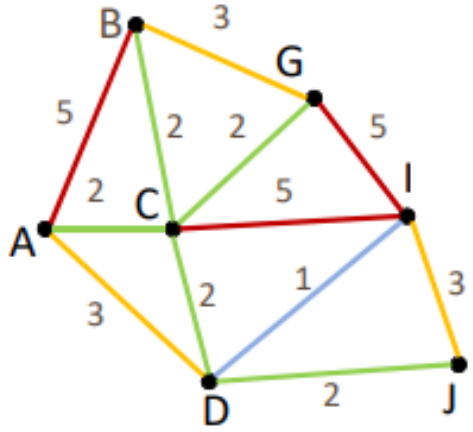
Closed:
{ B (0),
C (2) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```





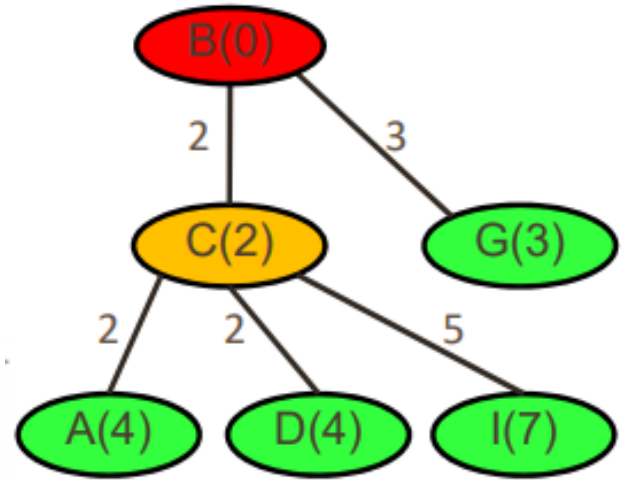
Open (Q):
{ G (3),
A (4),
D (4) }

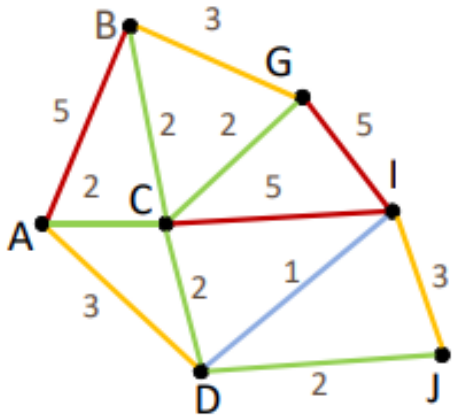
Closed:
{ B (0),
C (2) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

```
FORWARD_SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11          else
12              Resolve duplicate x'
13  return FAILURE
```



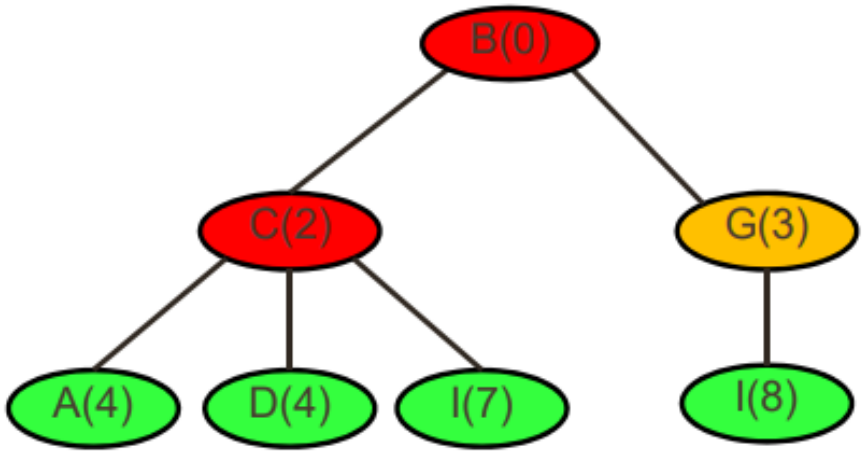


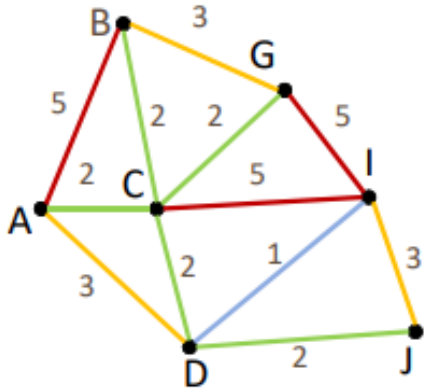
Open (Q):
{ G (3),
A (4),
D (4),
I (7) }

Closed:
{ B (0),
C (2) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam

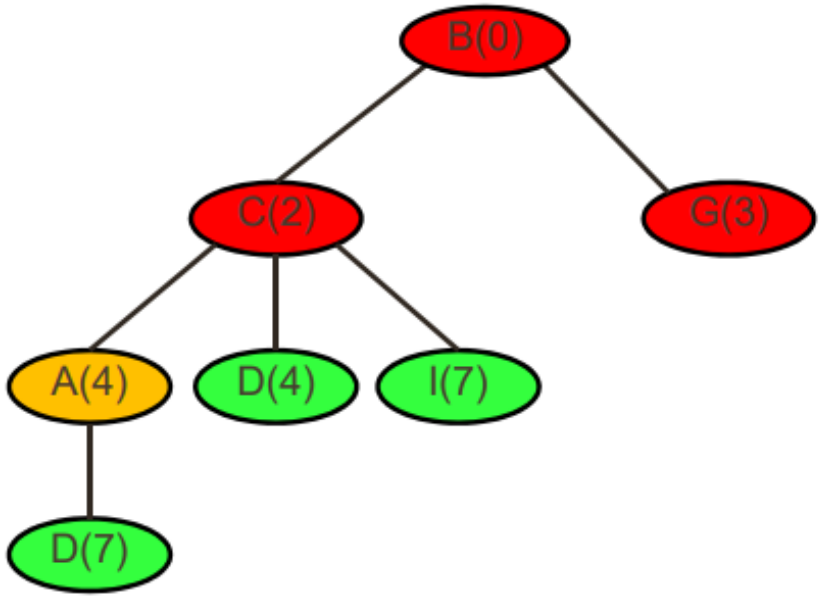


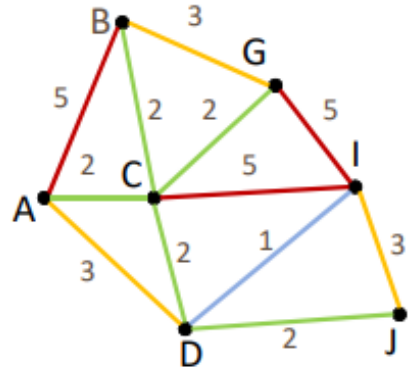


Open (Q):	Closed:
{ A (4),	{ B (0),
D (4),	C (2),
I (7) }	G (3) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam



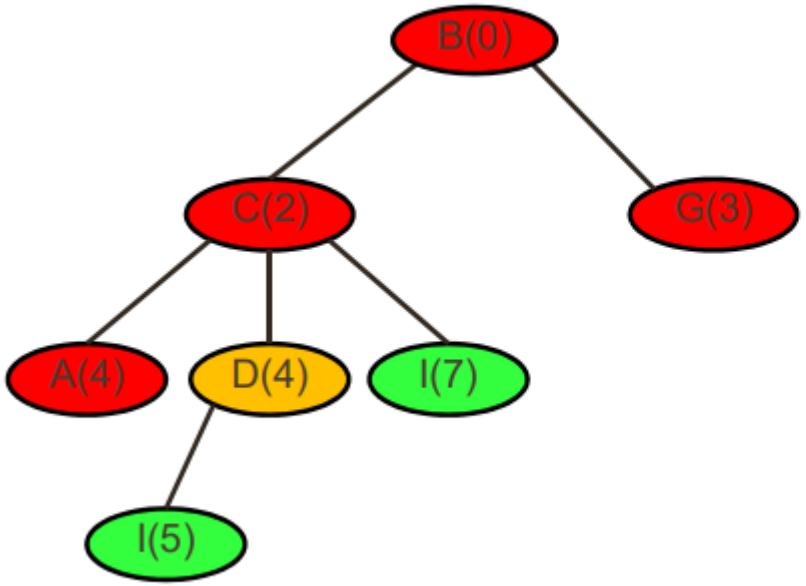


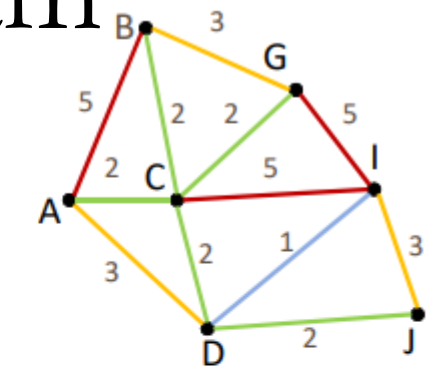
Open (Q):
{ D (4),
I (7) }

Closed:
{ B (0),
C (2),
G (3),
A (4) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam



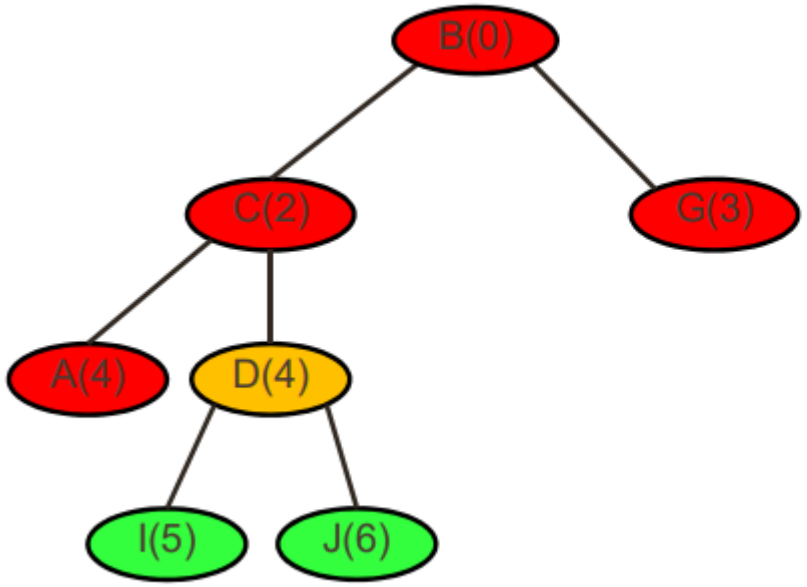


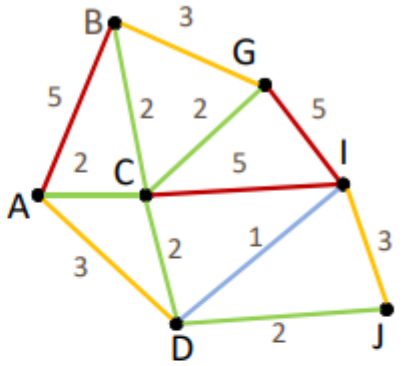
Open (Q):
{ I (5) }

Closed:
{ B (0),
C (2),
G (3),
A (4),
D (4) }

Co-funded by the
Erasmus+ Programme
of the European Union

Dijkstra's algoritam



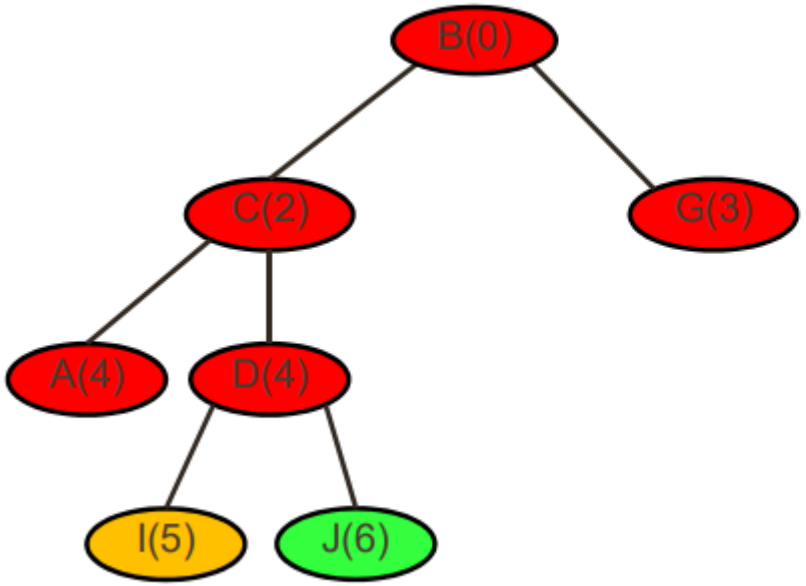


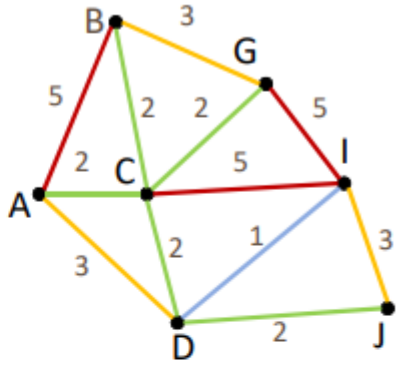
Open (Q):
{ I (5) ,
J (6) }

Closed:
{ B (0),
C (2),
G (3),
A (4),
D (4) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam



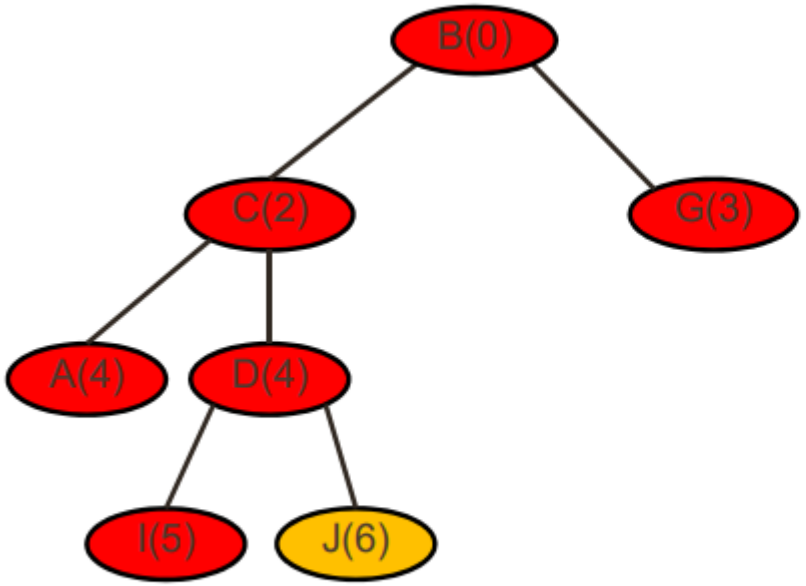


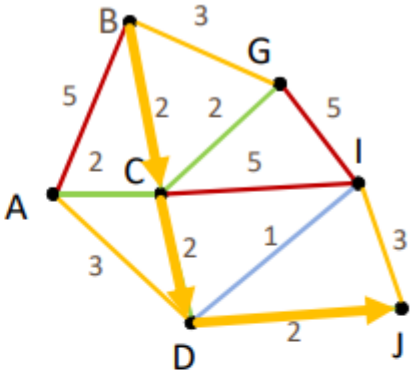
Open (Q):
{ J (6) }

Closed:
{ B (0),
C (2),
G (3),
A (4),
D (4),
I (5) }

Co-funded by the Erasmus+ Programme of the European Union

Dijkstra's algoritam





Open (Q):
{ }

Closed:
{ B (0),
C (2),
G (3),
A (4),
D (4),
I (5),
J (6) }

Final path solution: B→C→D→J
with path cost 6

Co-funded by the Erasmus+ Programme of the European Union

A* heuristička pretraga

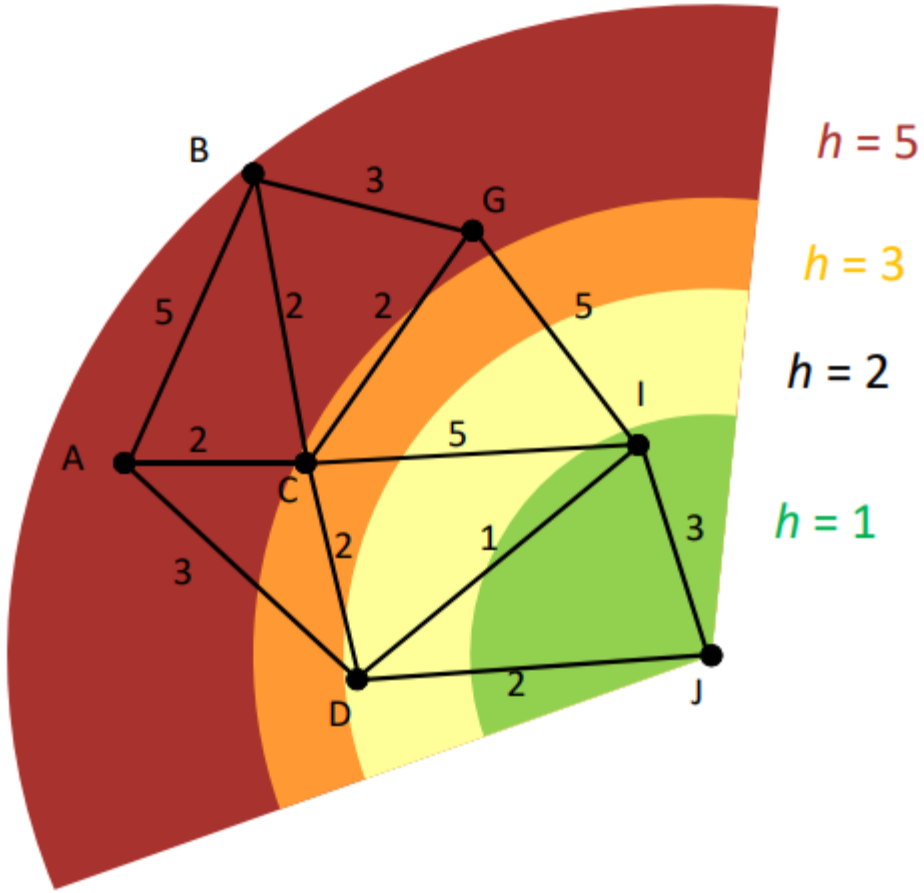
- Heuristika:
  - Funkcija koja opisuje koliko smo daleko od cilja
  - Uvek se postavlja za određeni problem
  - Menhetn ili Euklidska distanca, …
- A* funkcija cene: $f(n) = g(n) + h(n)$

Cena puta

Cena heuristike

Co-funded by the Erasmus+ Programme of the European Union

A* heuristička pretraga



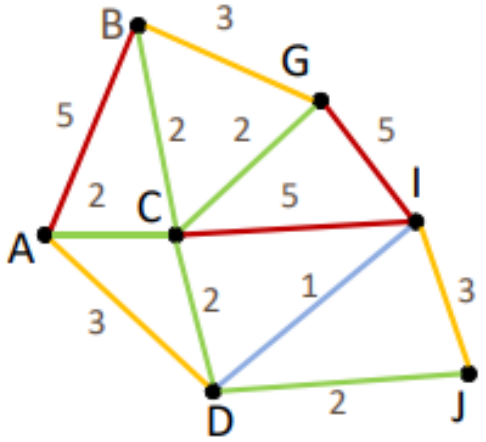
Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga

Open (Q): {B(0)}

Closed: {B(0)}

B(0)

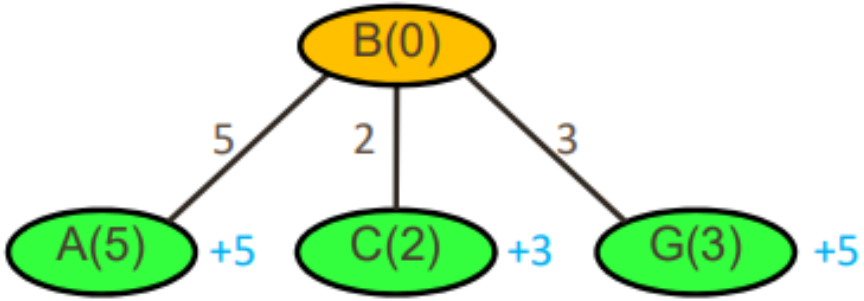


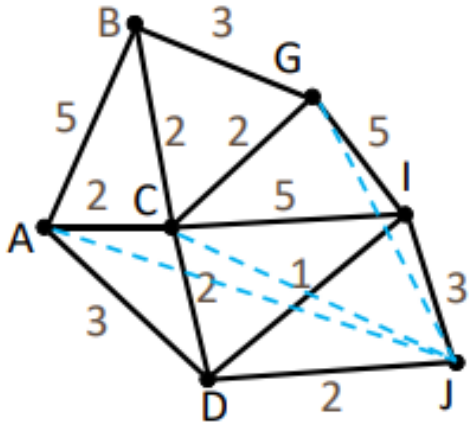
Our A* queue will be ordered by cost to arrive + heuristic:

- push (*Q.Insert*) by A* priority, *f(n)*
- pop (*Q.GetFirst*) from the front, and add it to the closed list

Co-funded by the Erasmus+ Programme of the European Union

A* heuristička pretraga



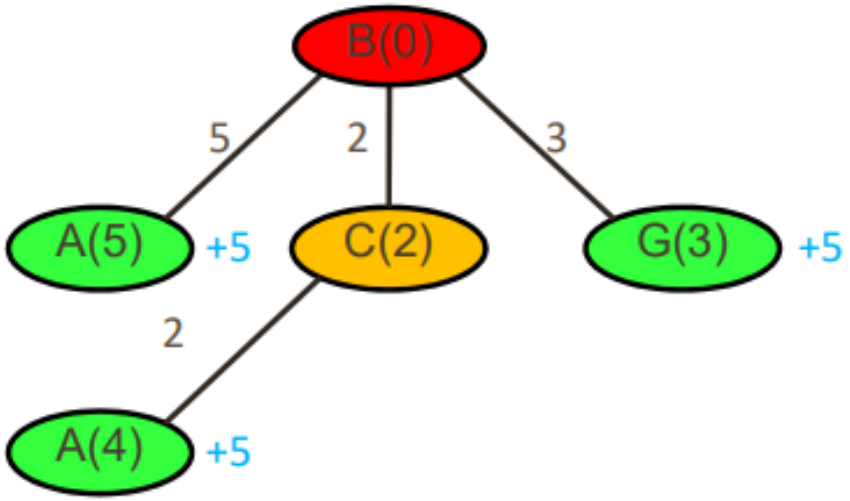


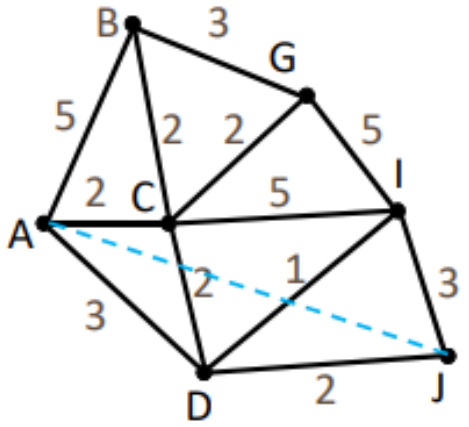
Open (Q):
{ C (2+3),
G (3+5),
A (5+5) }

Closed:
{ B (0) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



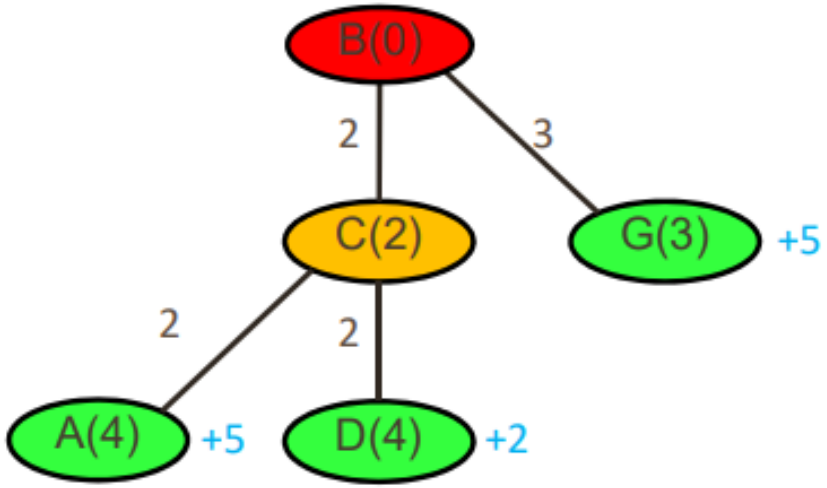


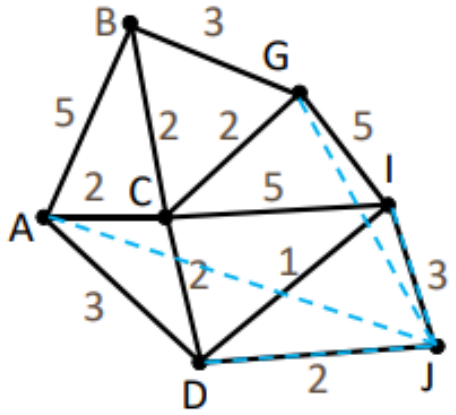
Open (Q):
{ G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



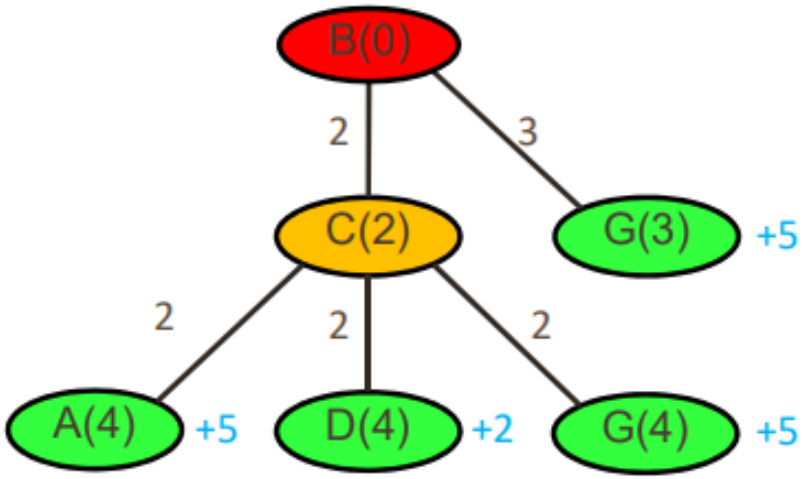


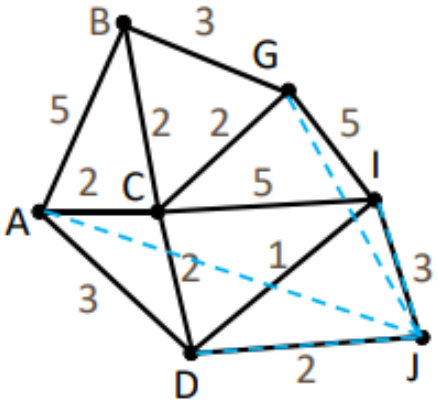
Open (Q):
{ D (4+2),
G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



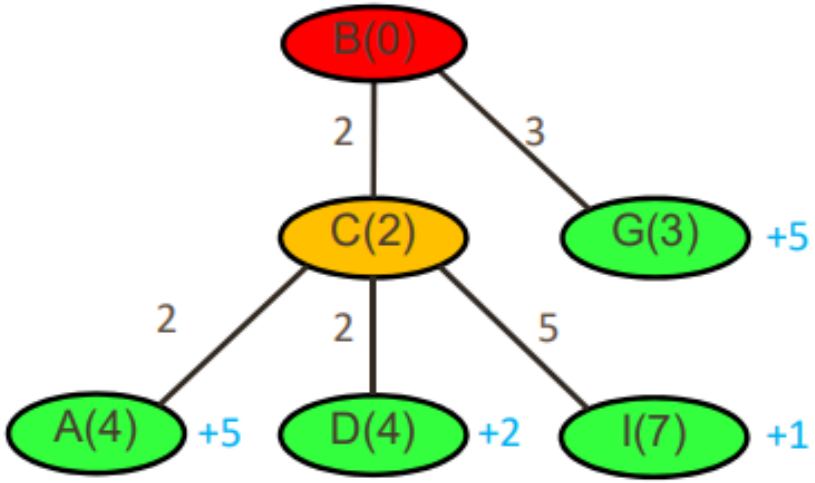


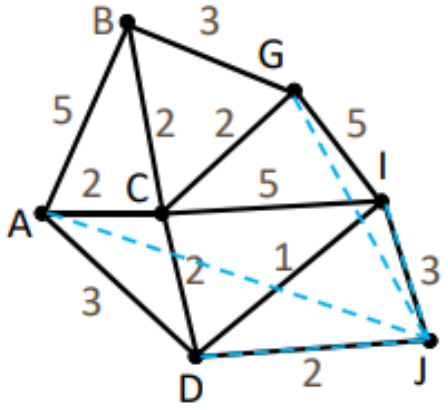
Open (Q):
{ D (4+2),
G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



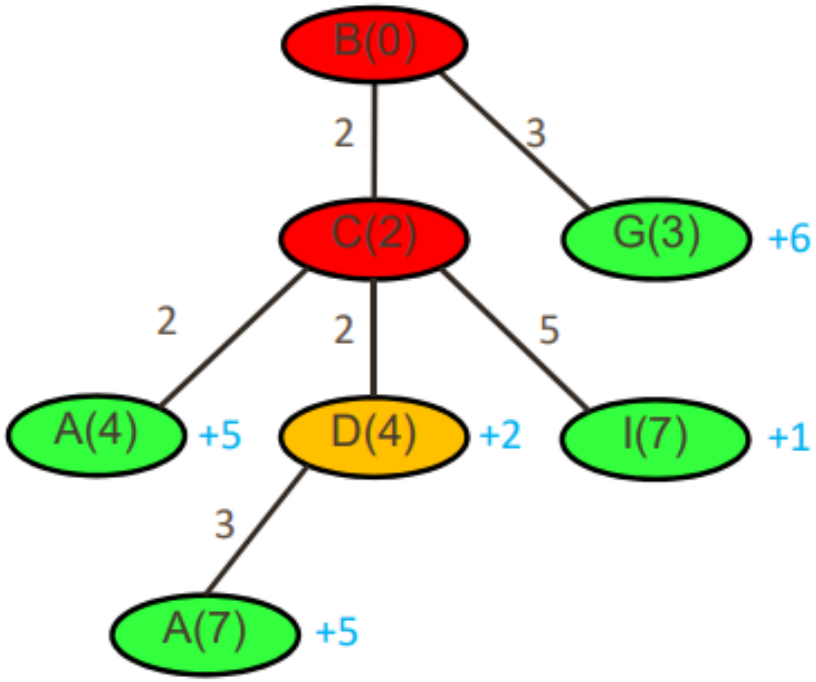


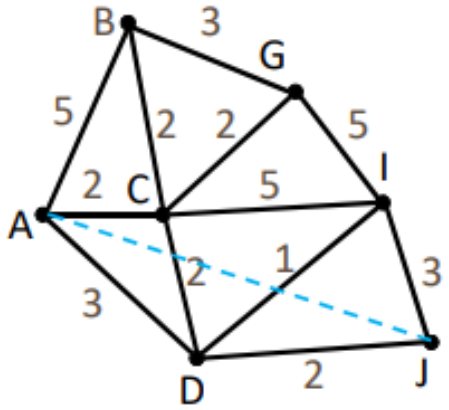
Open (Q):
{ D (4+2),
I (7+1),
G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



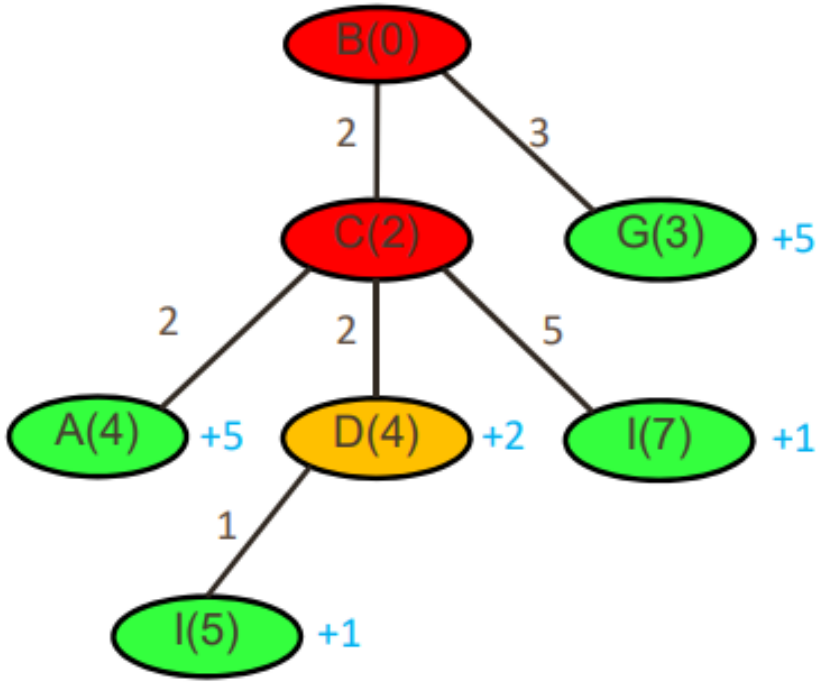


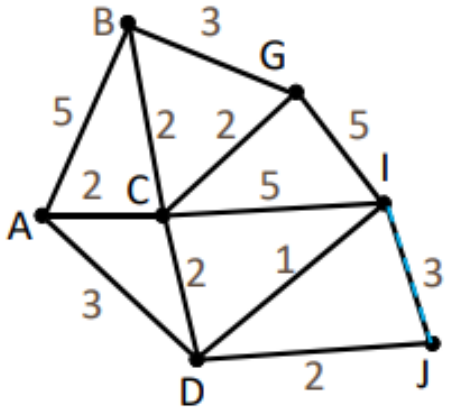
Open (Q):
{ I (5+1),
G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2),
D (4) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



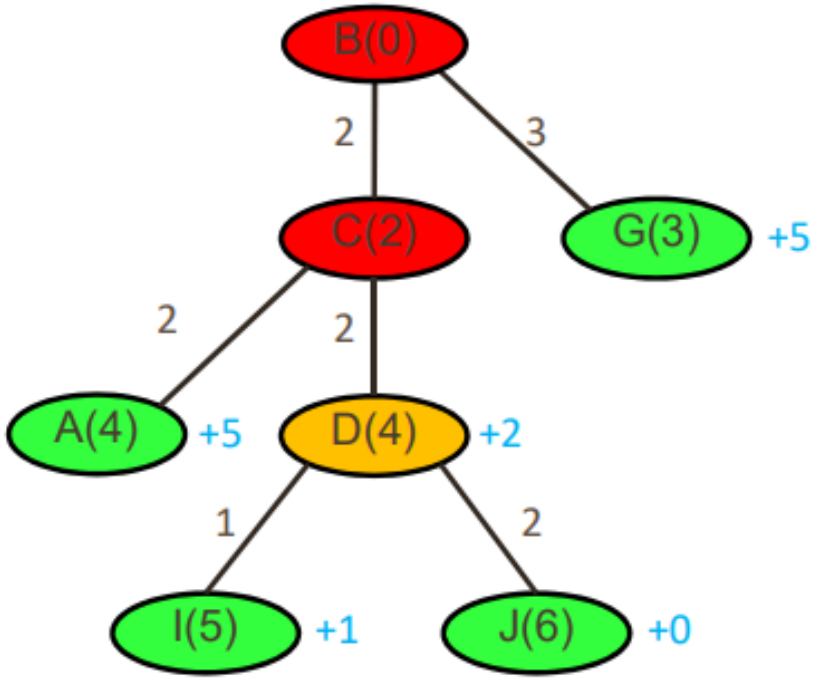


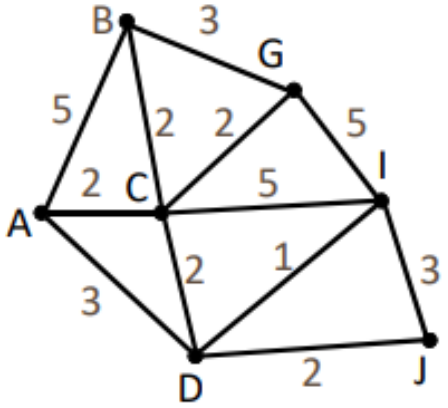
Open (Q):
{ I (5+1),
G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2),
D (4) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga



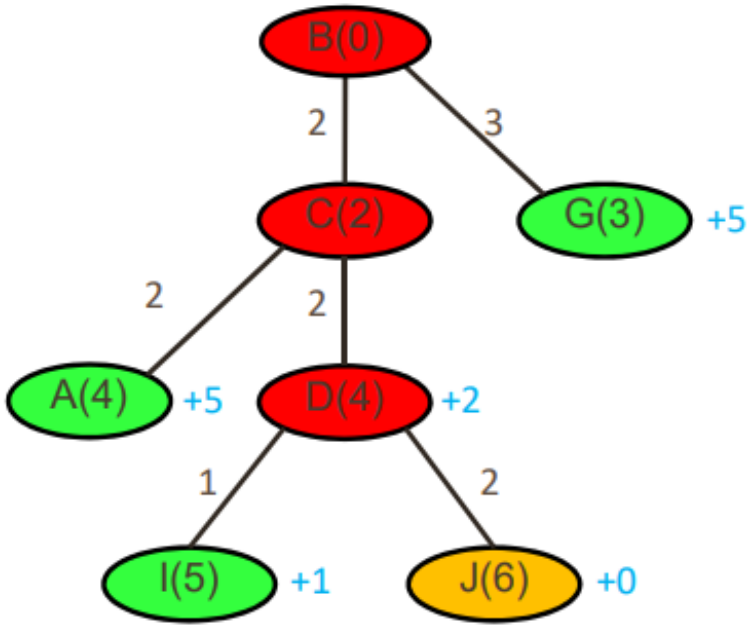


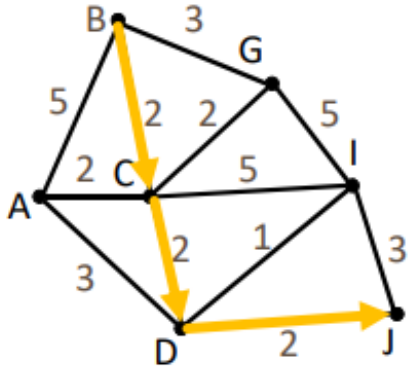
Open (Q):
{ J (6+0),
I (5+1),
G (3+5),
A (4+5) }

Closed:
{ B (0) ,
C (2),
D (4) }

Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga





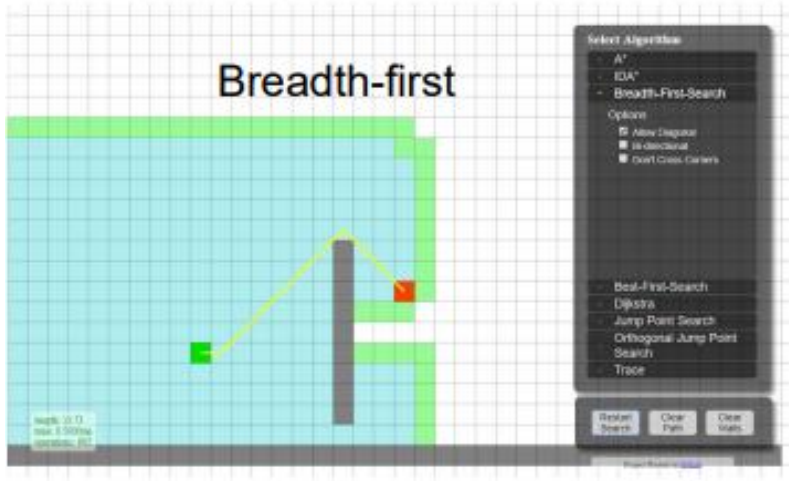
Open (Q):
{ I (5+1),
G (3+5),
A (4+5) }

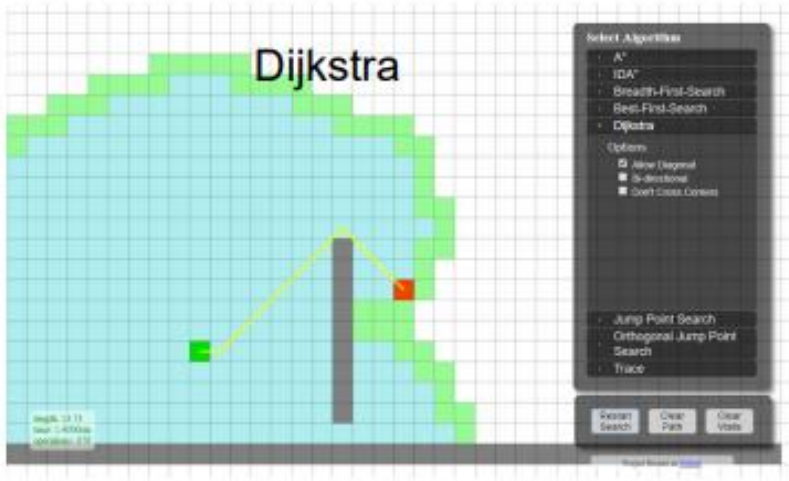
Closed:
{ B (0) ,
C (2),
D (4),
J (6) }

Final path solution: B→C→D→J
with path cost 6

Co-funded by the
Erasmus+ Programme
of the European Union

https://qiao.github.io/PathFinding.js/visual/





Co-funded by the Erasmus+ Programme of the European Union

Thanks!

Co-funded by the
Erasmus+ Programme
of the European Union