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

Theory of Robotics Systems

Graph Search

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

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

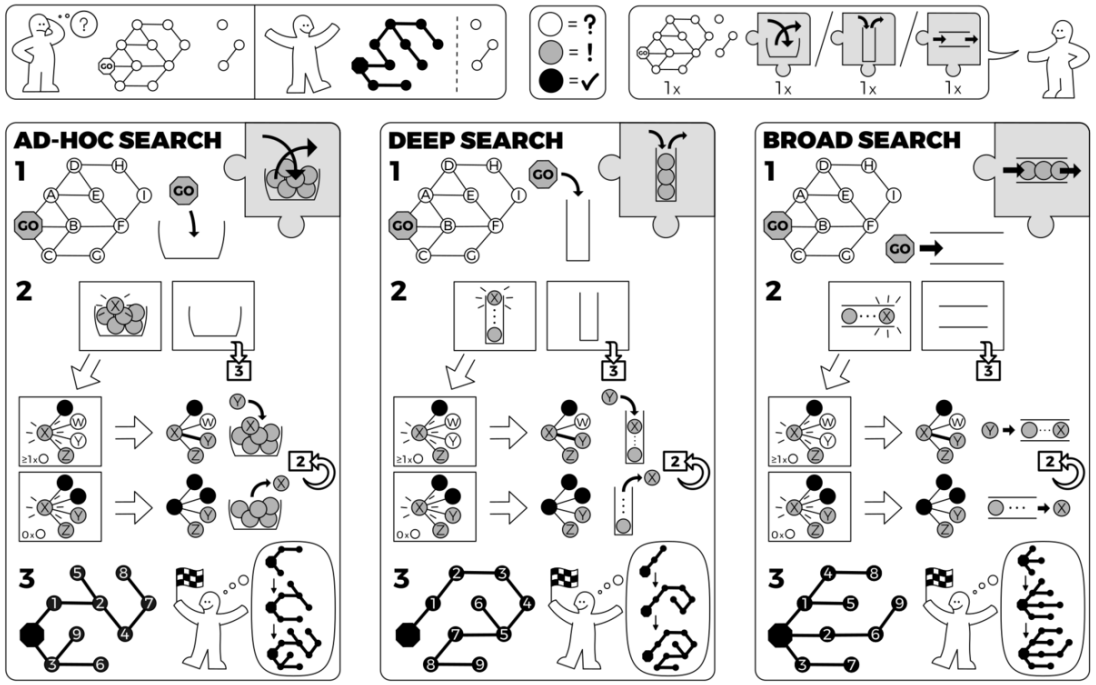
Zagreb University of Applied Sciences

School of Electrical Engineering
University of Belgrade

Faculty of Physics
Warsaw University of Technology

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

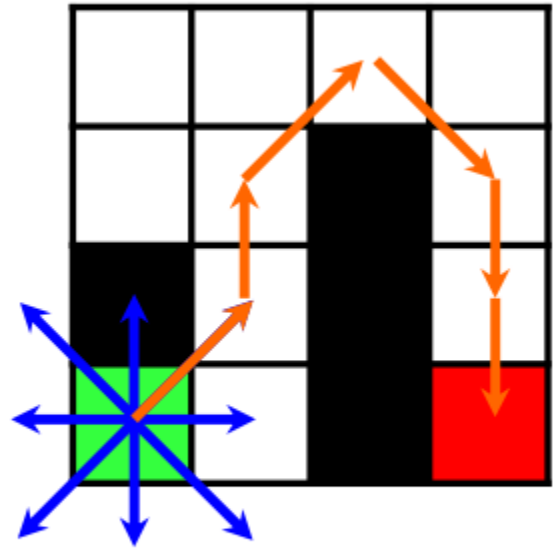
Metode pretrage grafova



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.

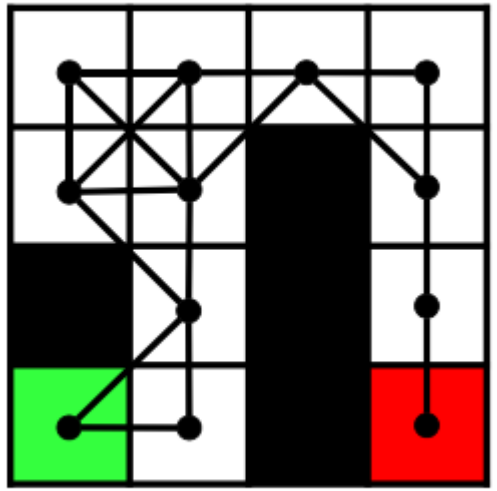


Assumptions about robot motion

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

Primer 1

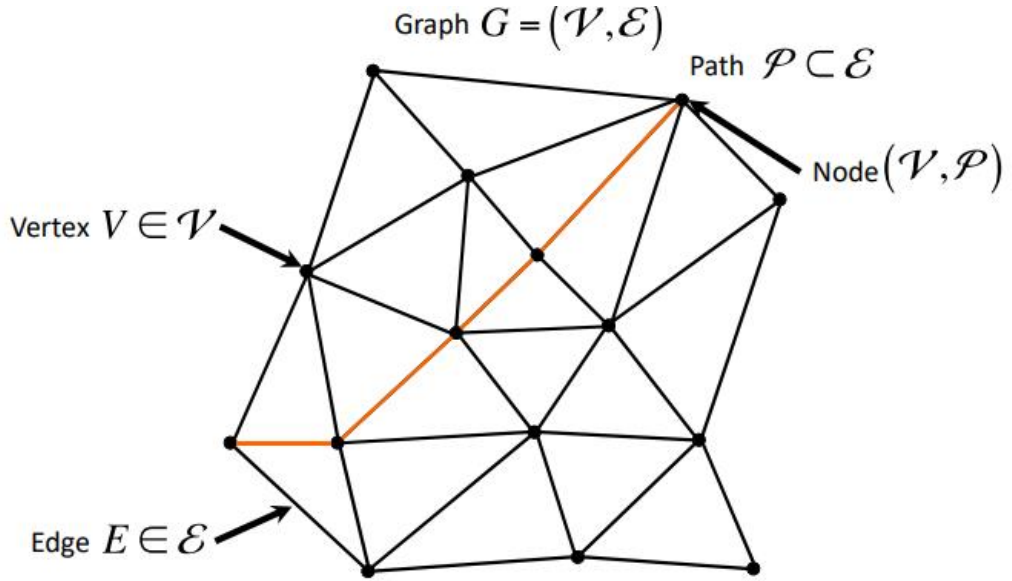
- Pretražimo dobijeni graf.
- Pretpostavljamo da sve ivice grafa mogu kreirati putanju.
- Rešimo problem tako što nađemo putanju od A do B.



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*

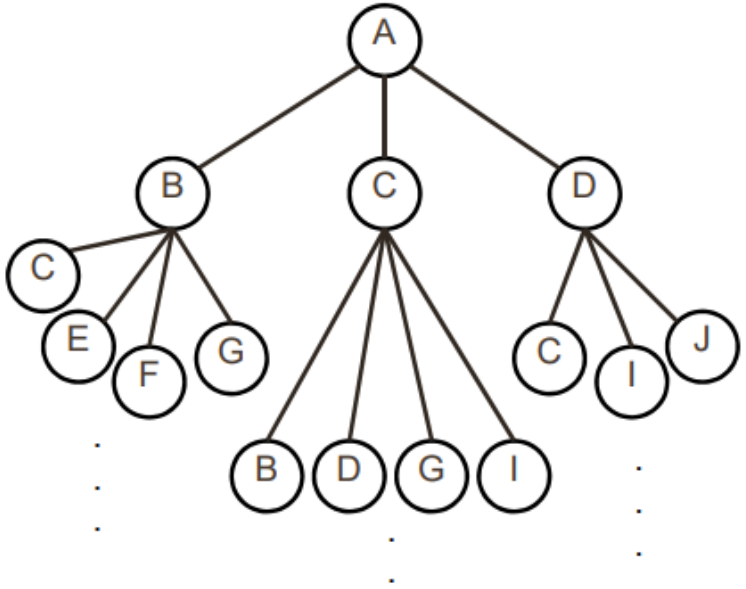


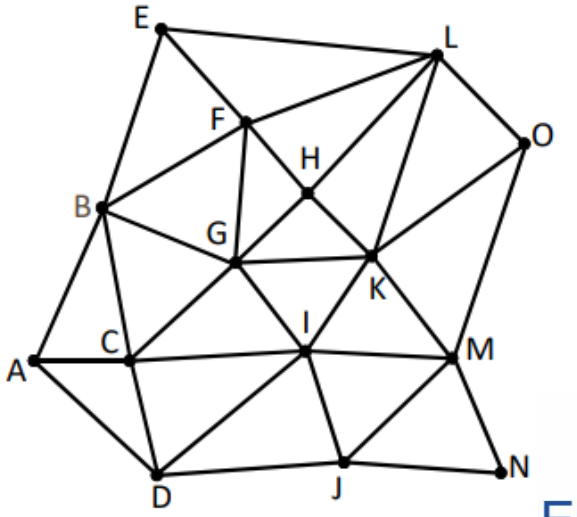
Directed graph: edges have direction
Weighted graph: edges have costs

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

Drvo pretrage

- Konstruišemo „drvo“ pomoću kojeg vršimo pretragu.

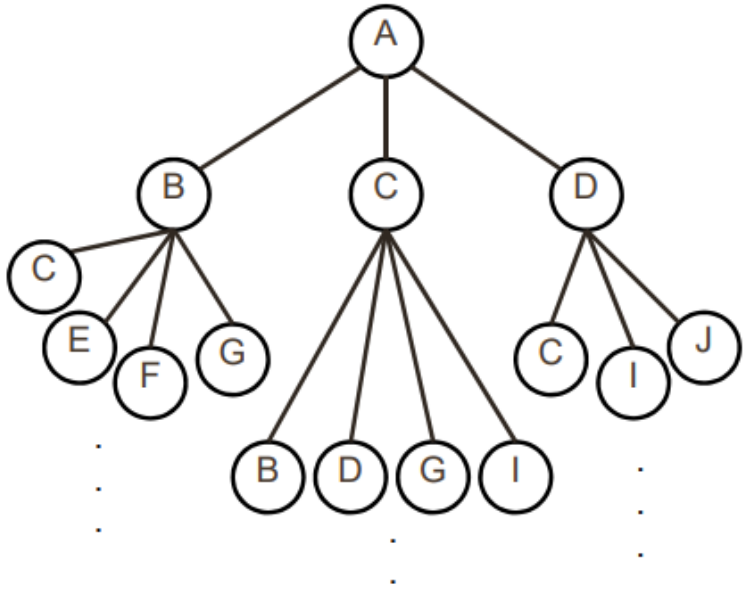




Co-funded by the Erasmus+ Programme of the European Union

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

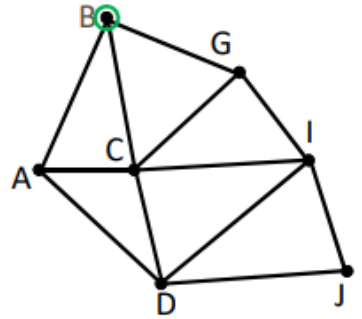
Osnove pravolinijske pretrage

```
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}

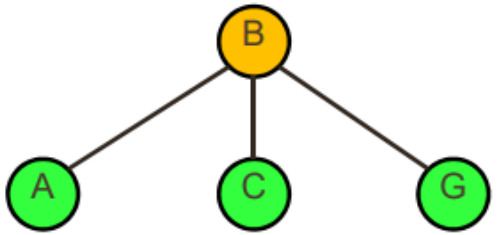
Closed: {}

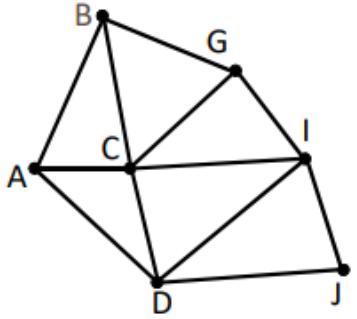
Our (BFS) queue will be FIFO:
- push (*Q.Insert*) onto the end
- pop (*Q.GetFirst*) from the front

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
```





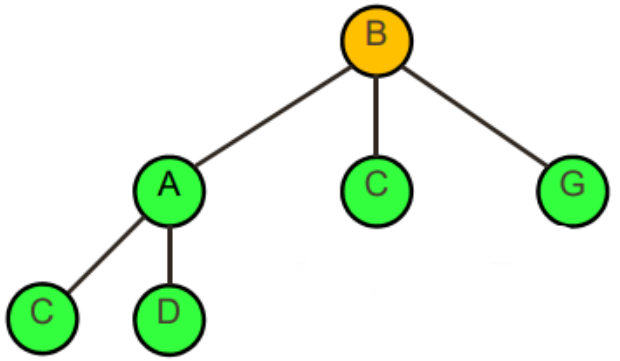
Open (Q): {A,C,G}

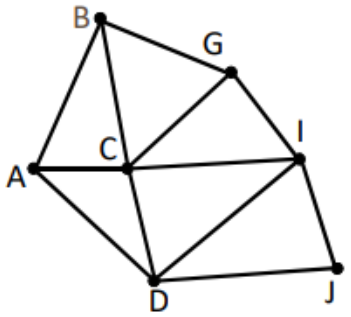
Closed: {B,A,C,G}

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
```



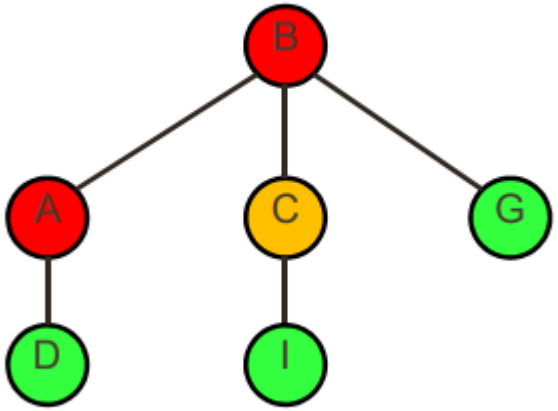


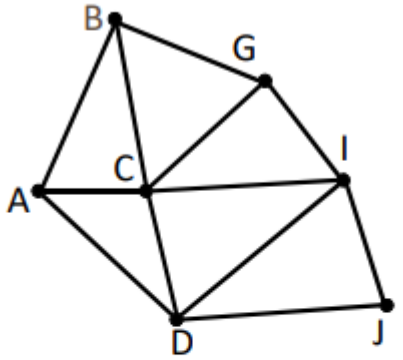
Open (Q): {C,G,D}

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

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

Pretraga po širini



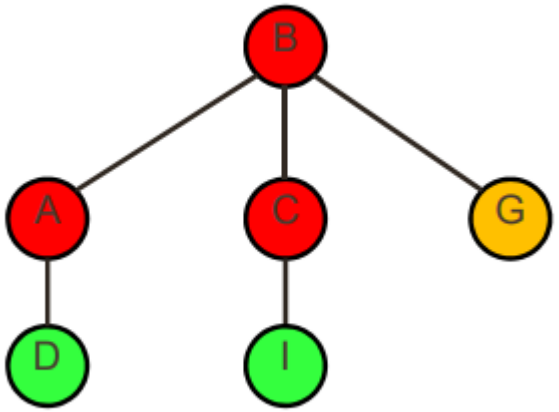


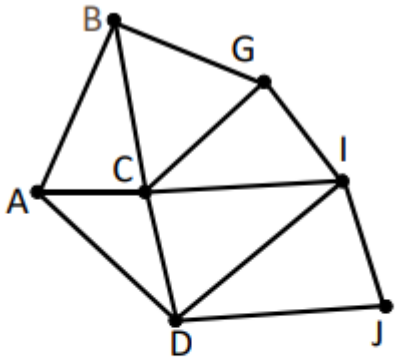
Open (Q): {G,D,I}

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

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

Pretraga po širini





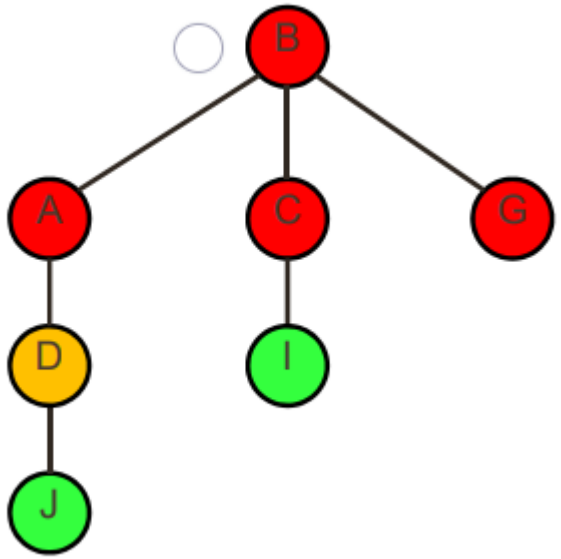
Open (Q): {D,I}

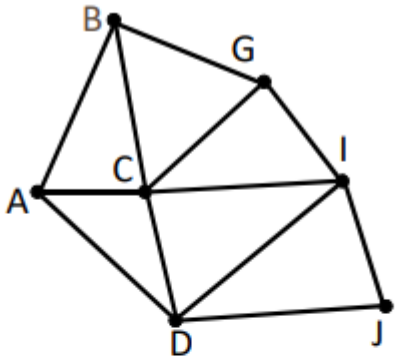
Closed: {B,A,C,G,D,I}

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

Itasdi

Pretraga po širini



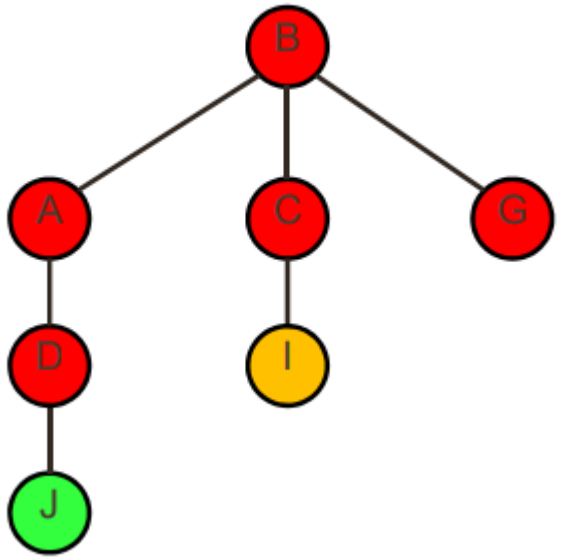


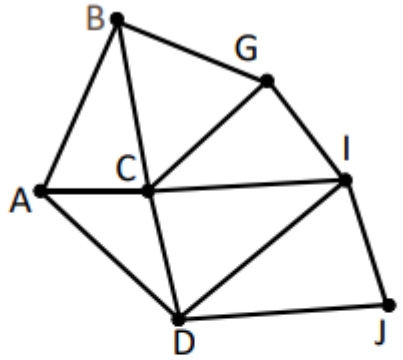
Open (Q): {I,J}

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

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

Pretraga po širini



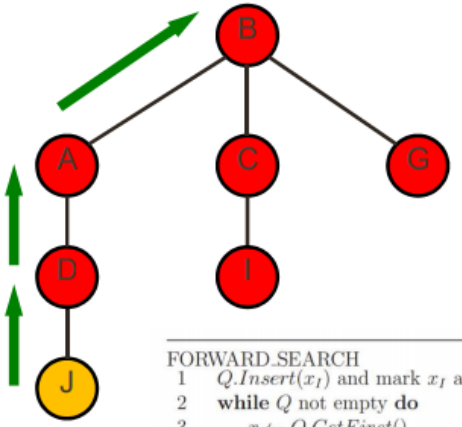


Open (Q):	Closed:
{J}	{B,A,C,G,D,I,J}

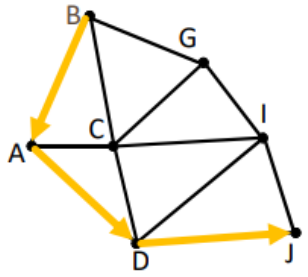
Co-funded by the Erasmus+ Programme of the European Union

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

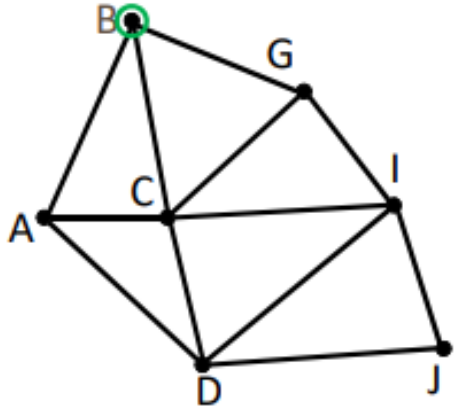
Other solutions may exist but have the same number or more transitions

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}

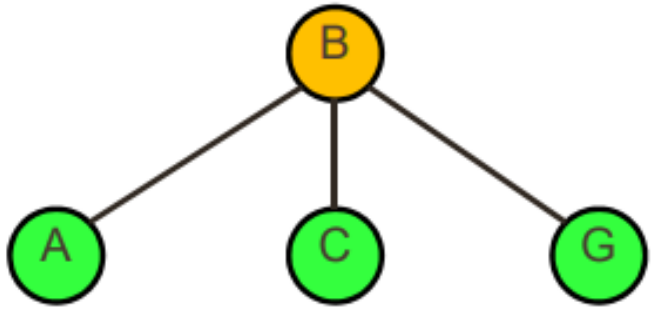
Closed: {}

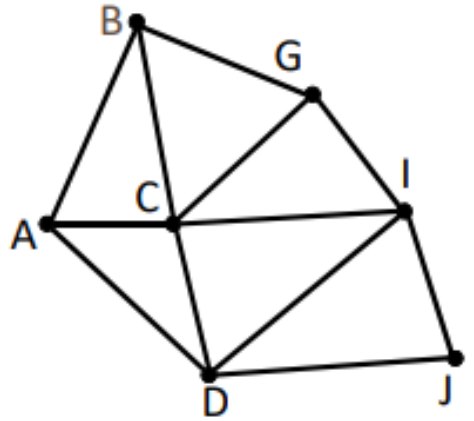
Our (DFS) queue will be LIFO:
- push (*Q.Insert*) onto the front
- pop (*Q.GetFirst*) from the front

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
```





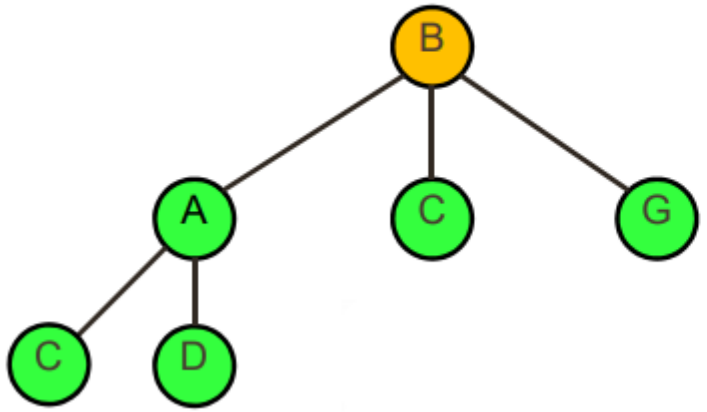
Open (Q): {A,C,G}

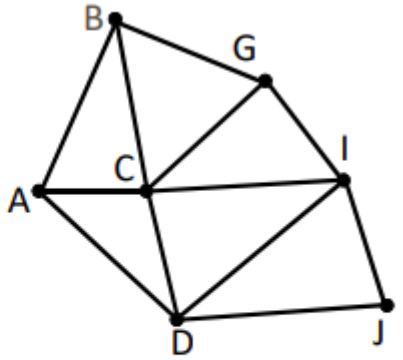
Closed: {B,A,C,G}

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
```



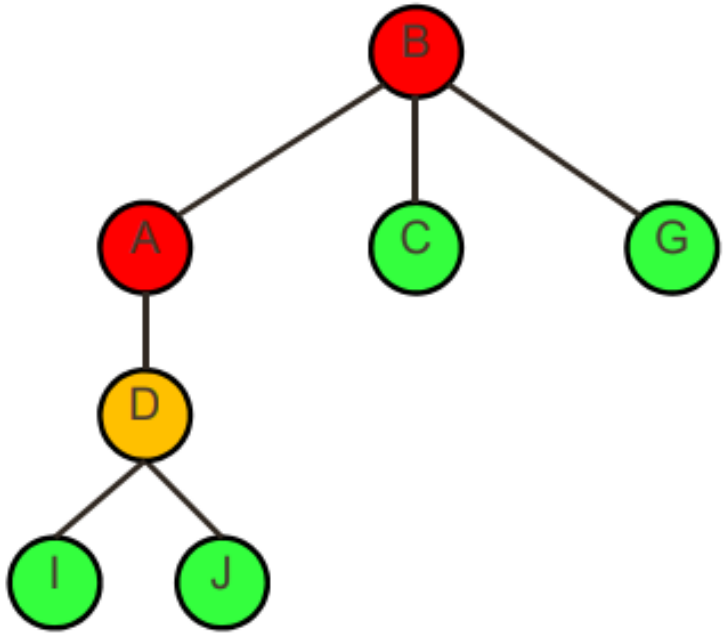


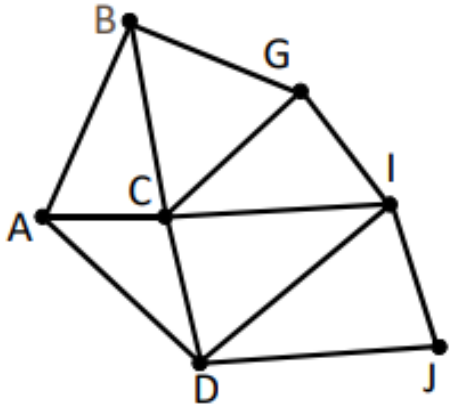
Open (Q): {D,C,G}

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

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

Pretraga po dubini



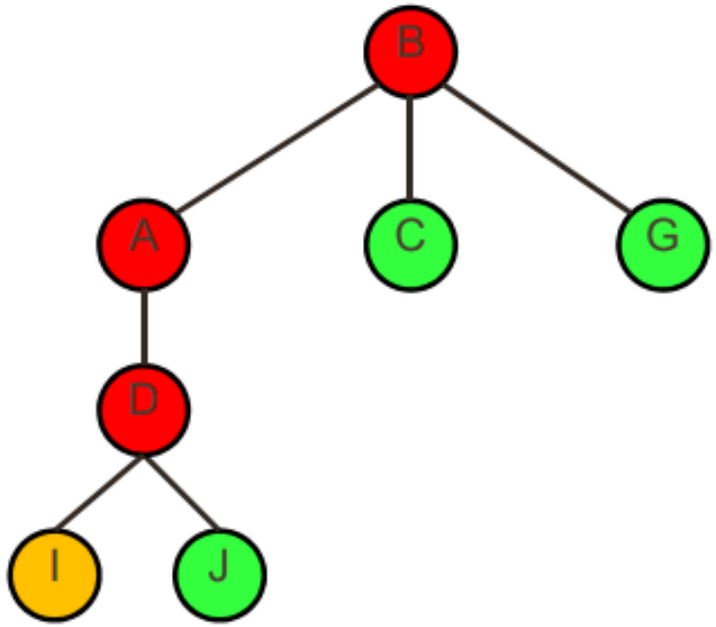


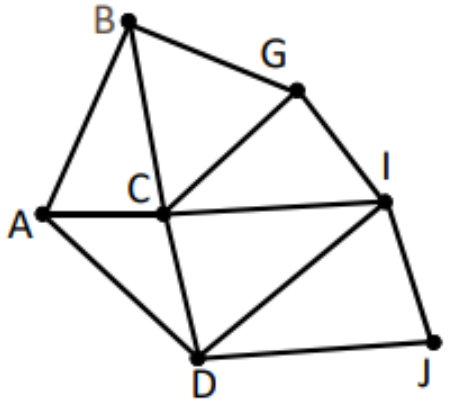
Open (Q): {I,J,C,G}

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

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

Pretraga po dubini





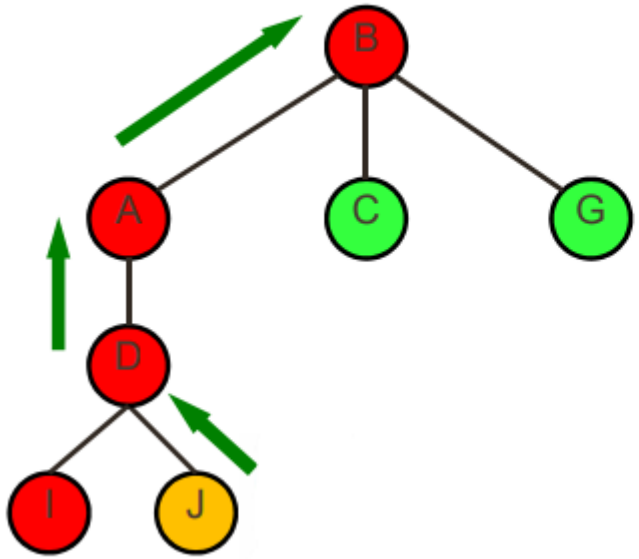
Open (Q): {J,C,G}

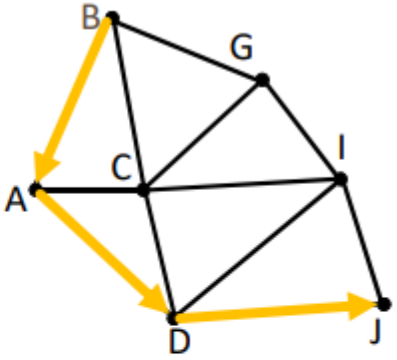
Closed: {B,A,C,G,D,I,J}

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
```





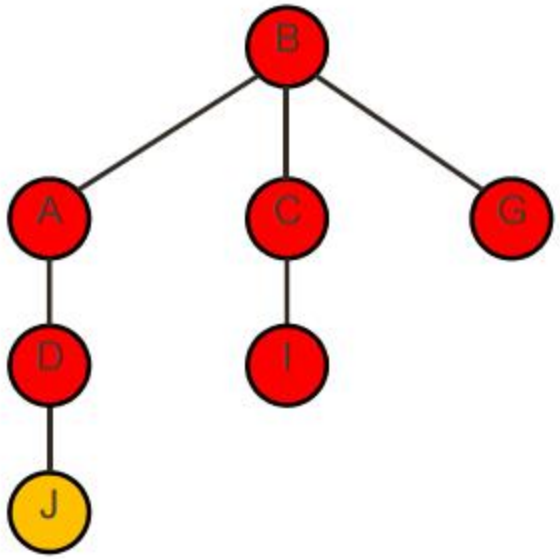
Open (Q): {}

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

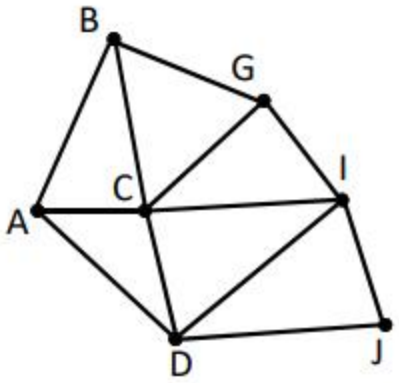
Final path solution: B→A→D→J

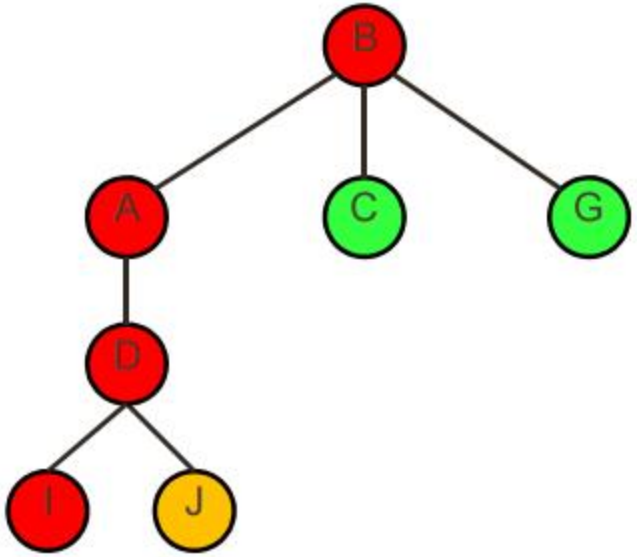
Co-funded by the Erasmus+ Programme of the European Union

Uporedna analiza



BFS



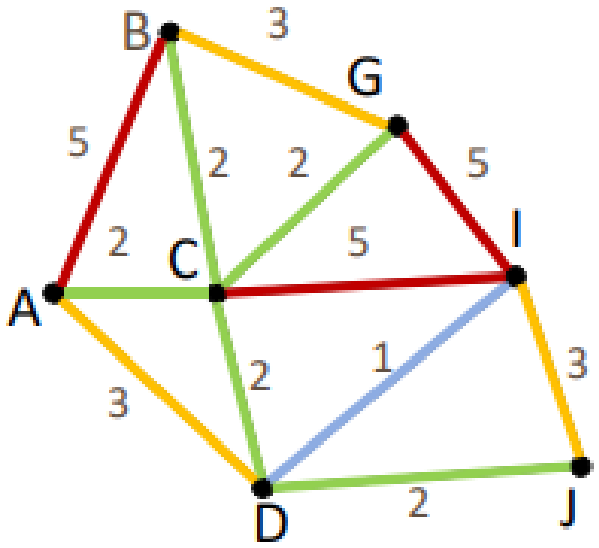


DFS

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

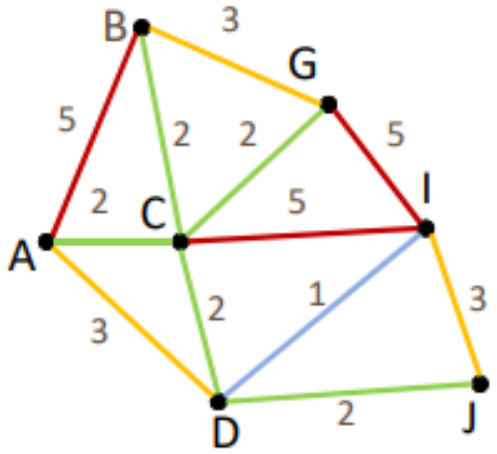
```
FORWARD SEARCH
1   Q.Insert(x_I) and mark x_I as visited
2   while Q not empty do
3       x ← Q.GetFirst()
4       if x ∈ X_G
5           return SUCCESS
6       forall u ∈ U(x)
7           x' ← f(x, u)
8           if x' not visited
9               Mark x' as visited
10              Q.Insert(x')
11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

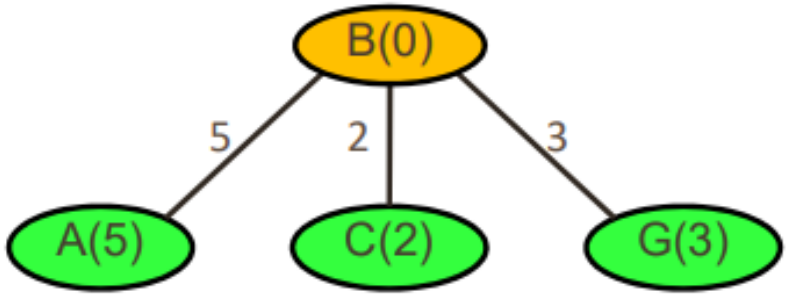
- push (*Q.Insert*) by cost
- pop (*Q.GetFirst*) from the front, and add it to the closed list

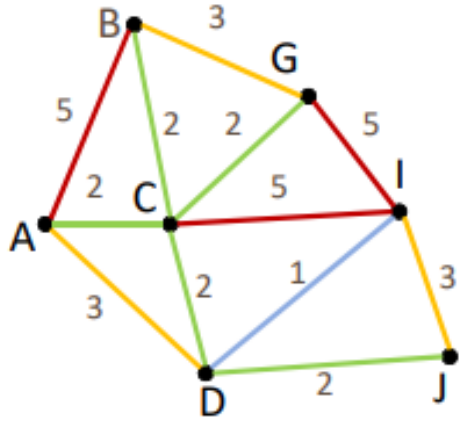


Co-funded by the Erasmus+ Programme of the European Union

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
```





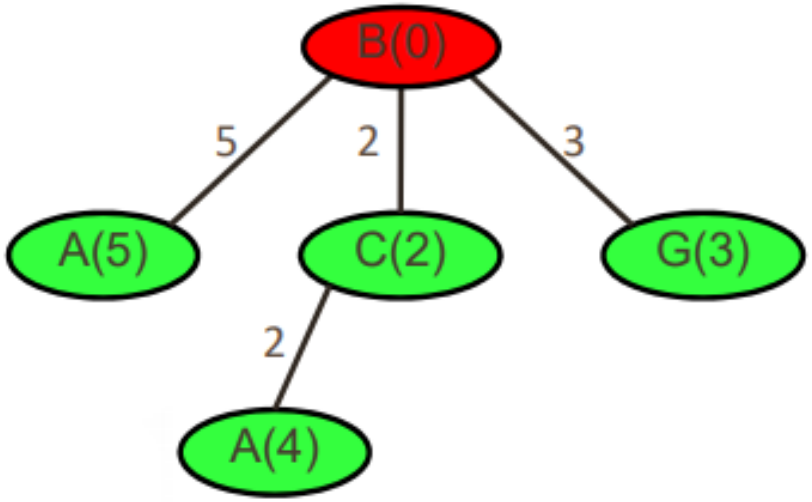
Open (Q):
{ C (2),
G (3),
A (5) }

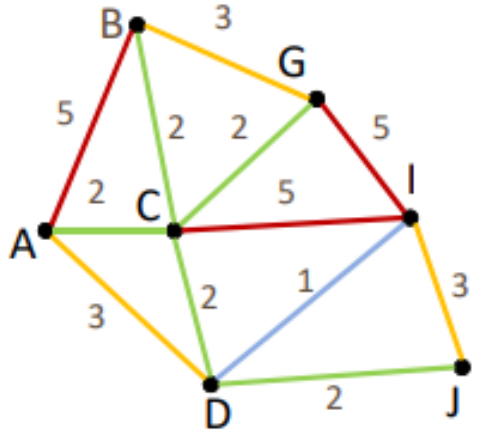
Closed:
{ B (0) }

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
```





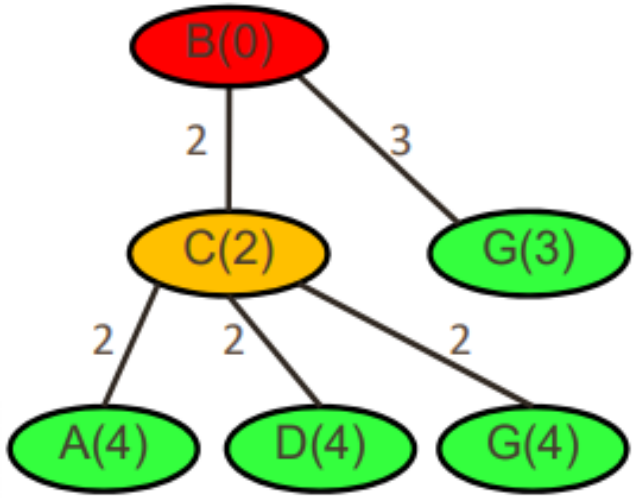
Open (Q):
{ G (3),
A (4) }

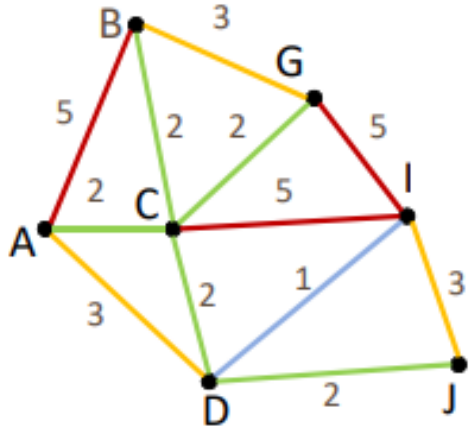
Closed:
{ B (0),
C (2) }

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
```





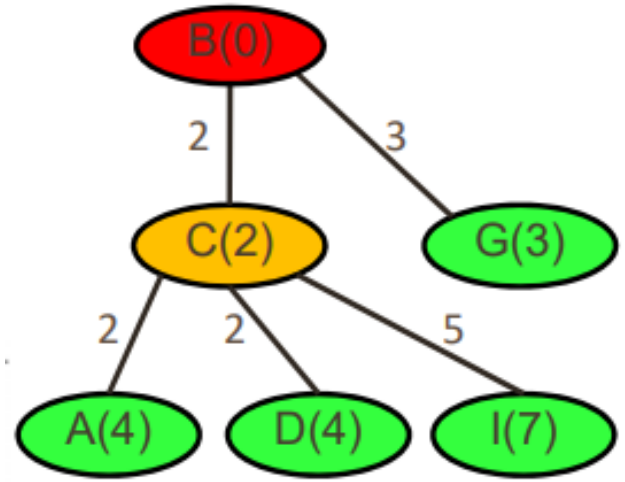
Open (Q):
{ G (3),
A (4),
D (4) }

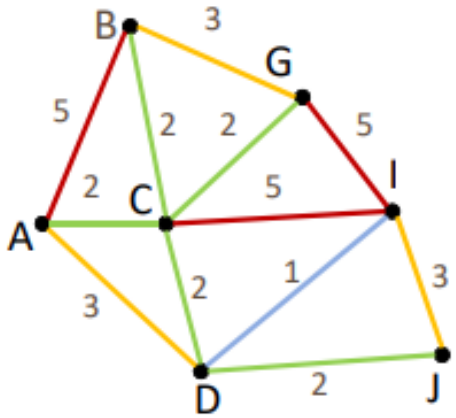
Closed:
{ B (0),
C (2) }

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
```



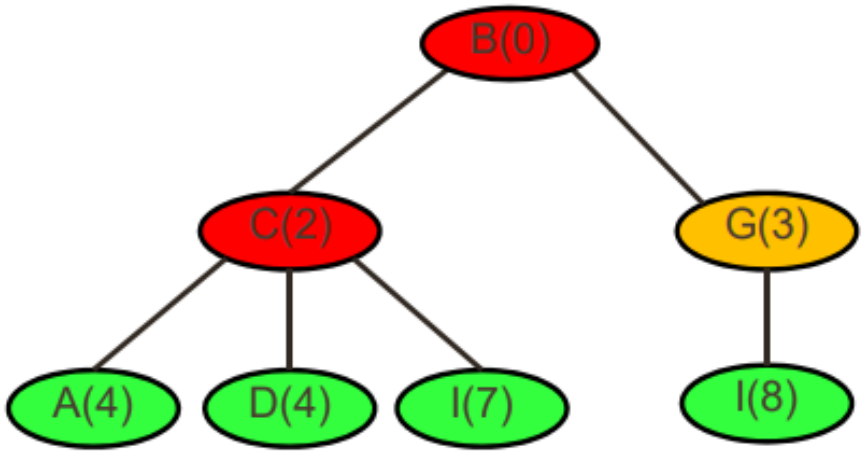


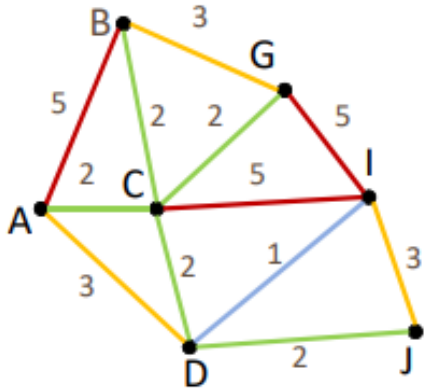
Open (Q):
{ G (3),
A (4),
D (4),
I (7) }

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

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

Dijkstra's algoritam

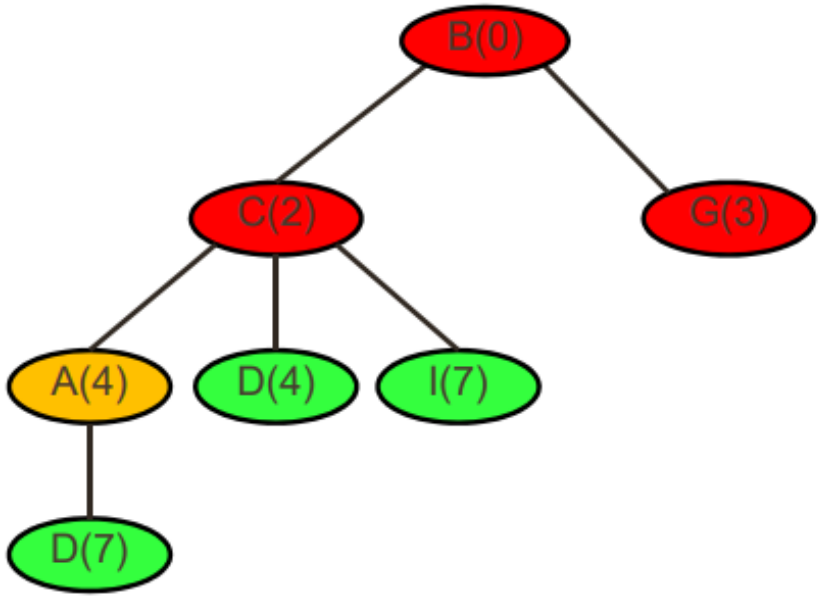


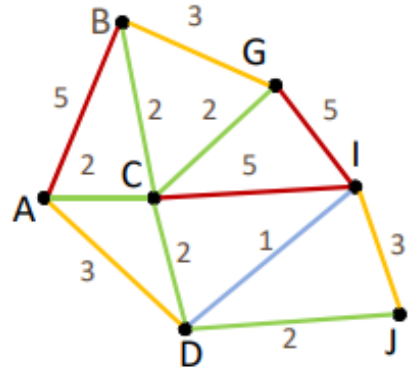


Open (Q):	Closed:
{ A (4),	{ B (0),
D (4),	C (2),
I (7) }	G (3) }

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

Dijkstra's algoritam



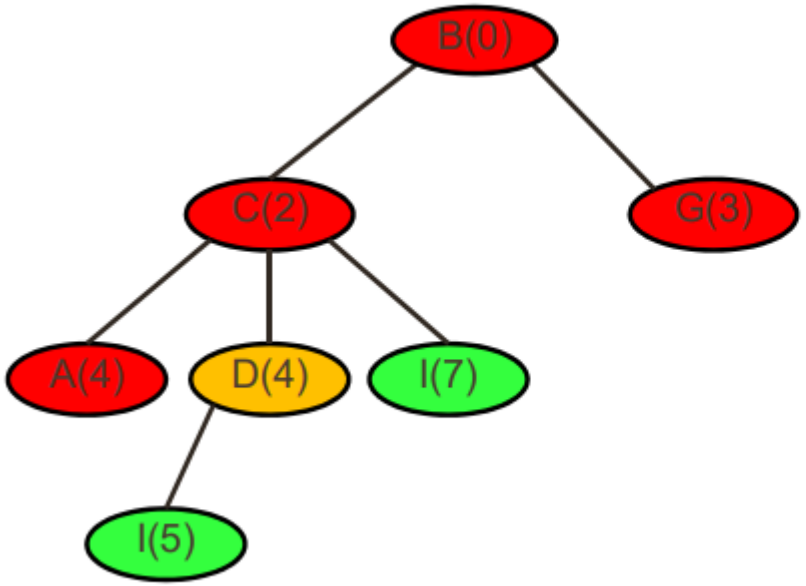


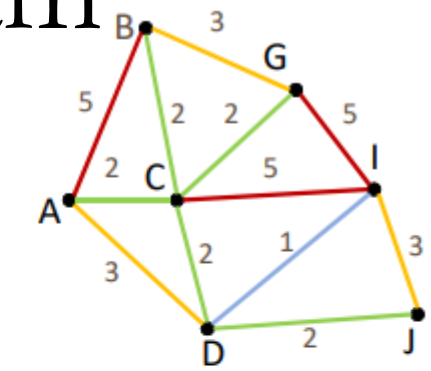
Open (Q):
{ D (4),
I (7) }

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

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

Dijkstra's algoritam



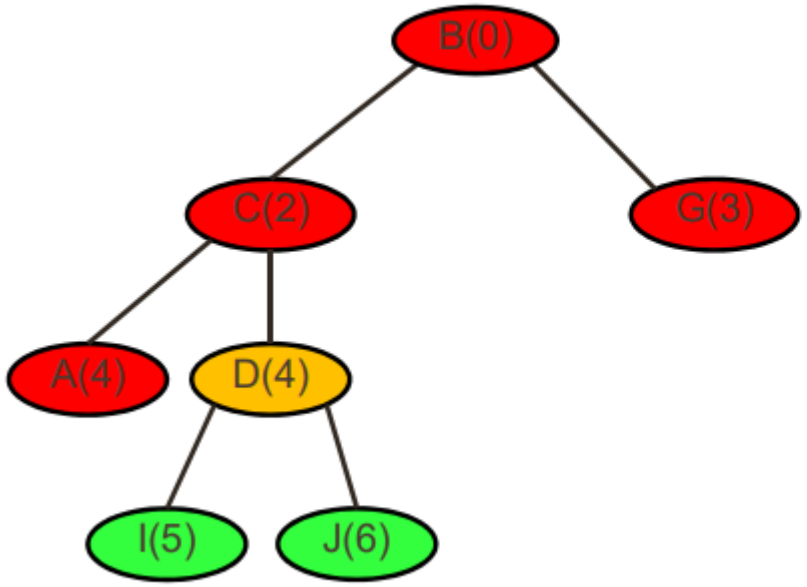


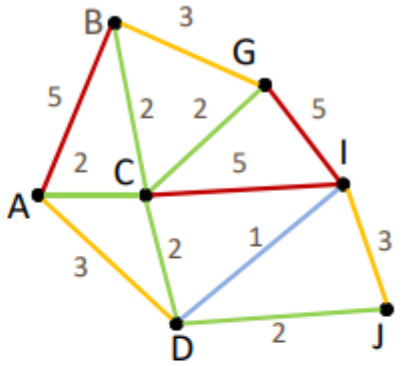
Open (Q):
{ I (5) }

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

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

Dijkstra's algoritam



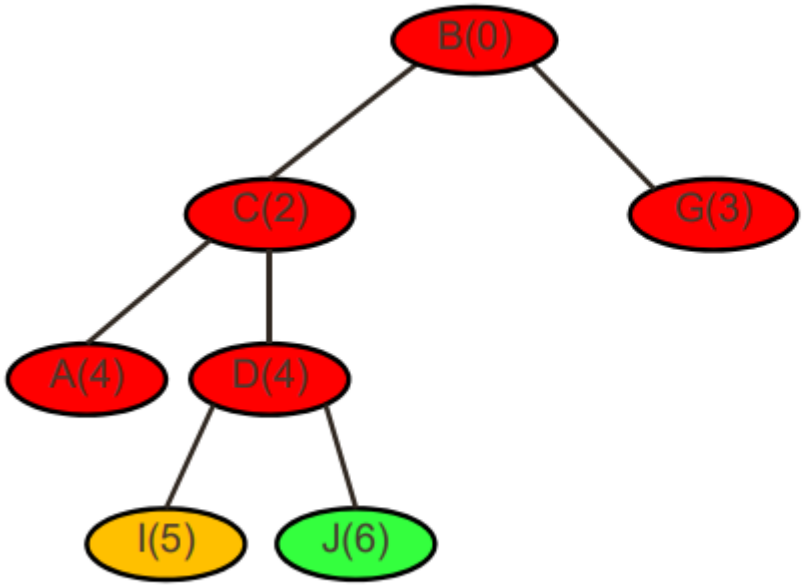


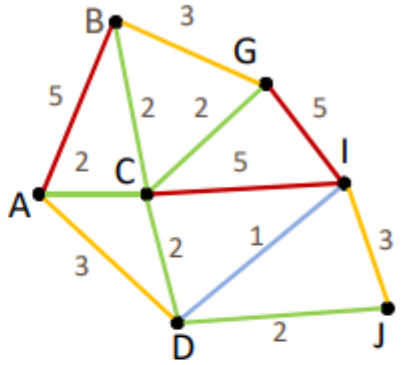
Open (Q):
{ I (5) ,
J (6) }

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

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

Dijkstra's algoritam



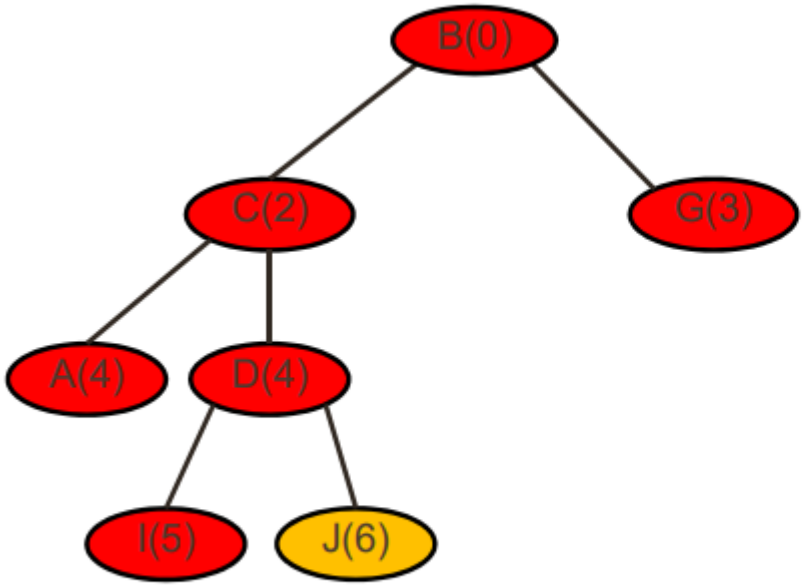


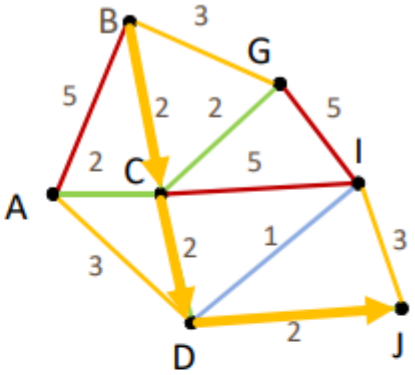
Open (Q):
{ J (6) }

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

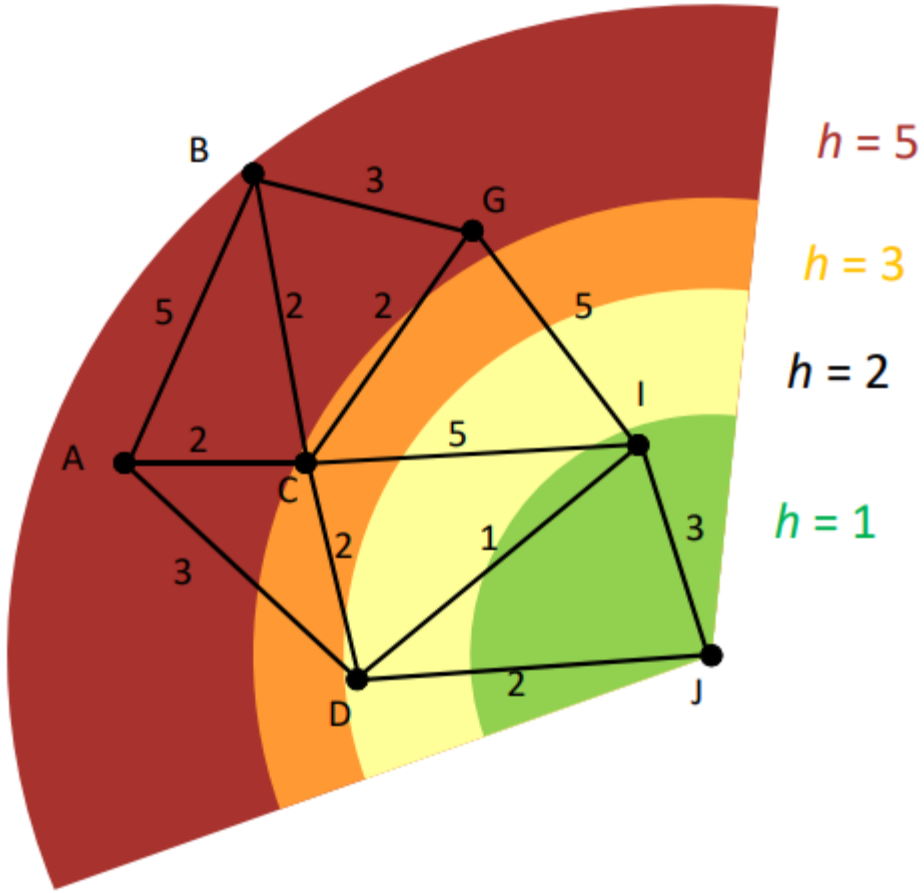
- Heuristika:
  - Funkcija koja opisuje koliko smo daleko od cilja
  - Uvek se postavlja za određeni problem
  - Menhetn ili Euklidska distanca, …
- A* funkcija cene: $f(n) = g(n) + h(n)$

Cena puta

Cena heuristike

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

A* heuristička pretraga



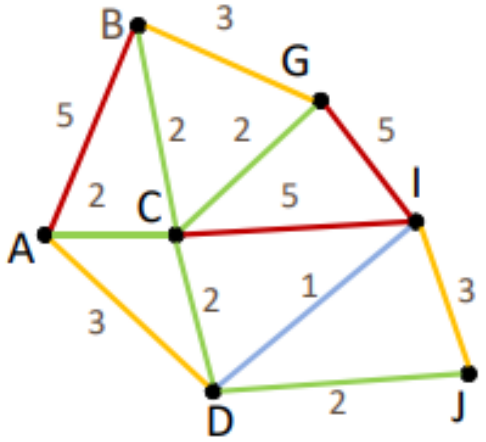
Co-funded by the
Erasmus+ Programme
of the European Union

A* heuristička pretraga

Open (Q): {B(0)}

Closed: {B(0)}

B(0)

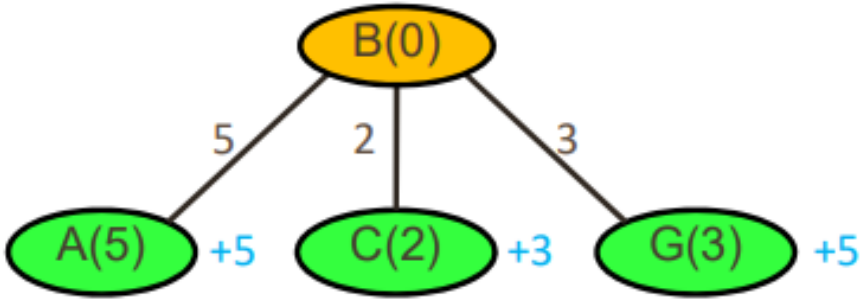


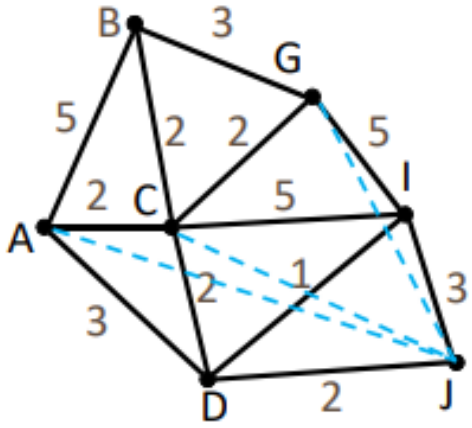
Our A* queue will be ordered by cost to arrive + heuristic:

- 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



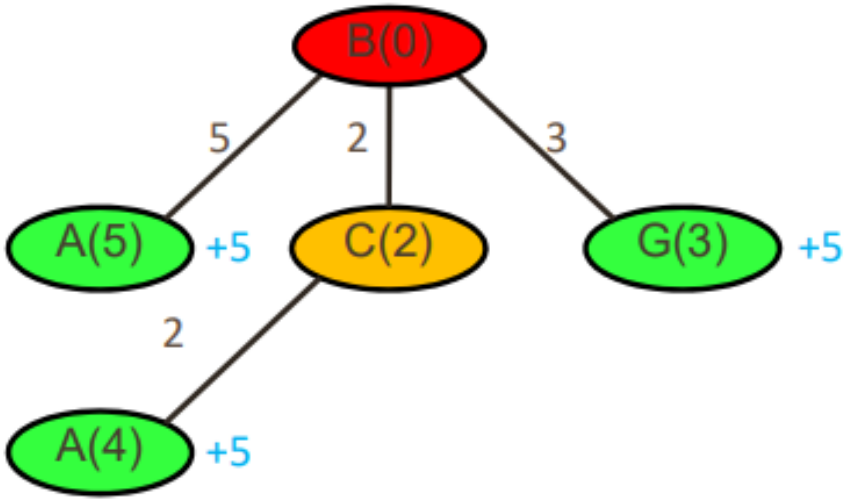


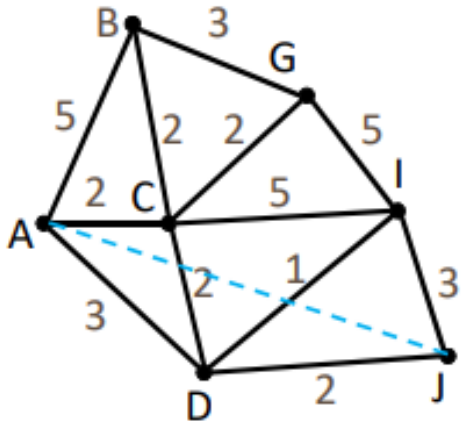
Open (Q):
{ C (2+3),
G (3+5),
A (5+5) }

Closed:
{ B (0) }

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

A* heuristička pretraga



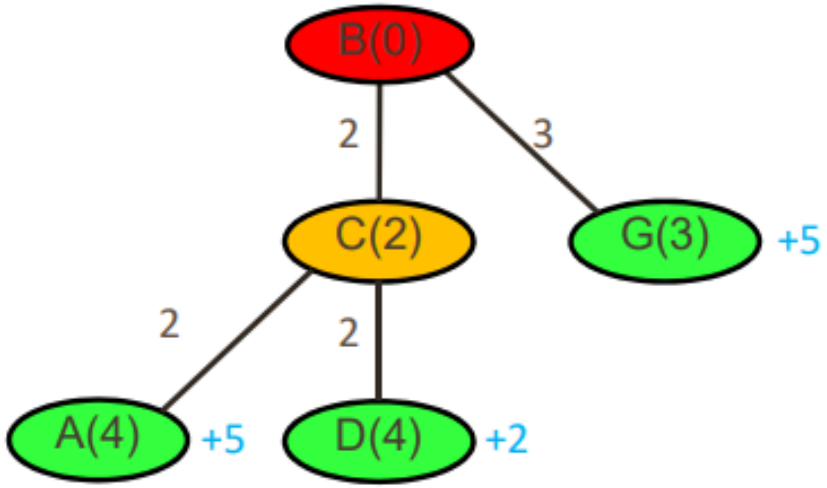


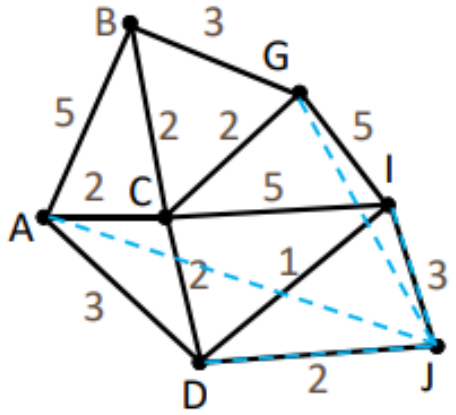
Open (Q):
{ G (3+5),
A (4+5) }

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

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

A* heuristička pretraga



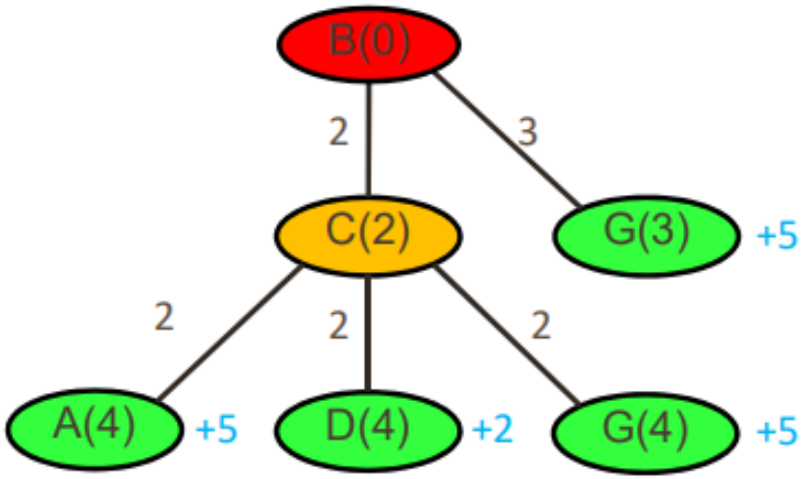


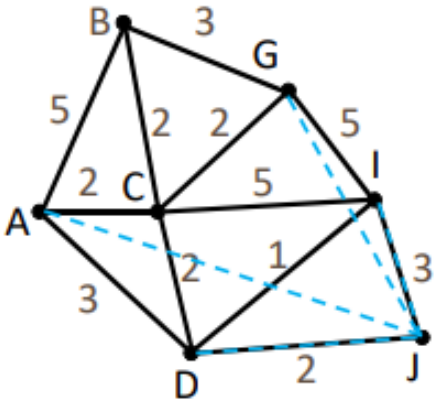
Open (Q):
{ D (4+2),
G (3+5),
A (4+5) }

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

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

A* heuristička pretraga



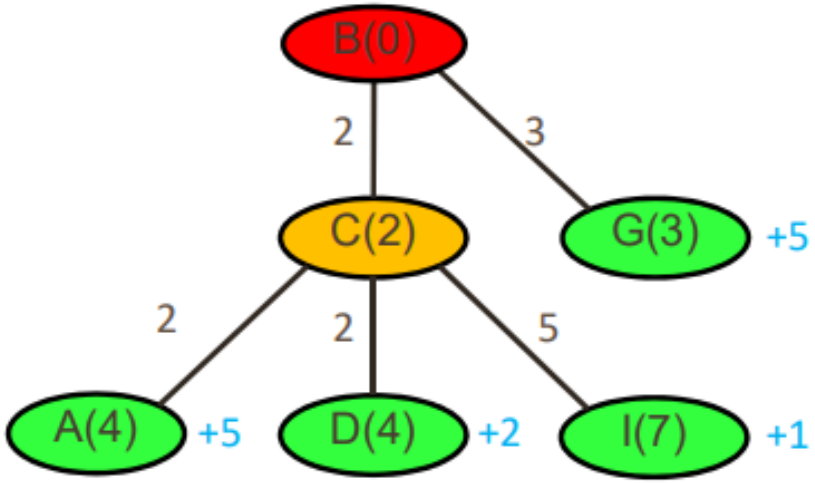


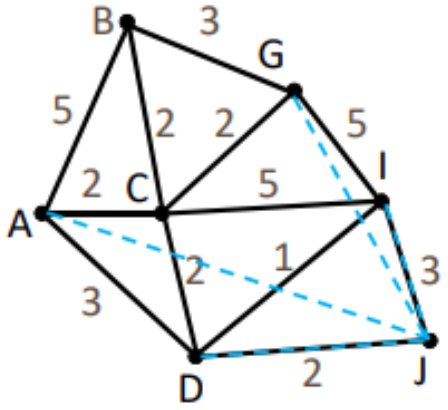
Open (Q):
{ D (4+2),
G (3+5),
A (4+5) }

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

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

A* heuristička pretraga



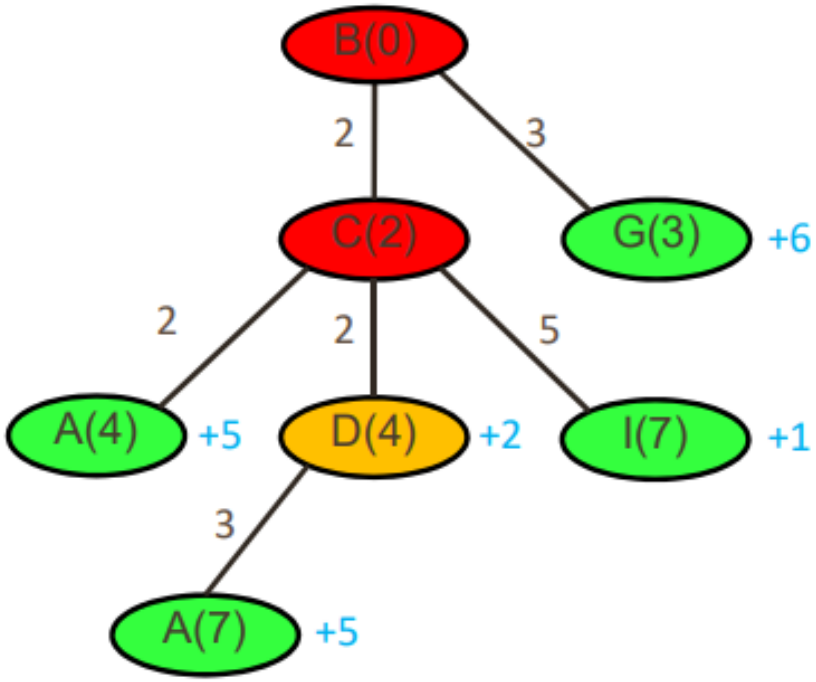


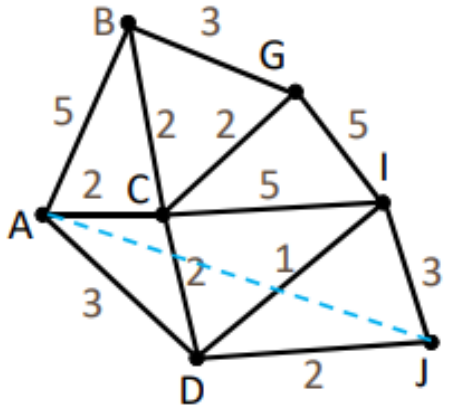
Open (Q):
{ D (4+2),
I (7+1),
G (3+5),
A (4+5) }

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

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

A* heuristička pretraga



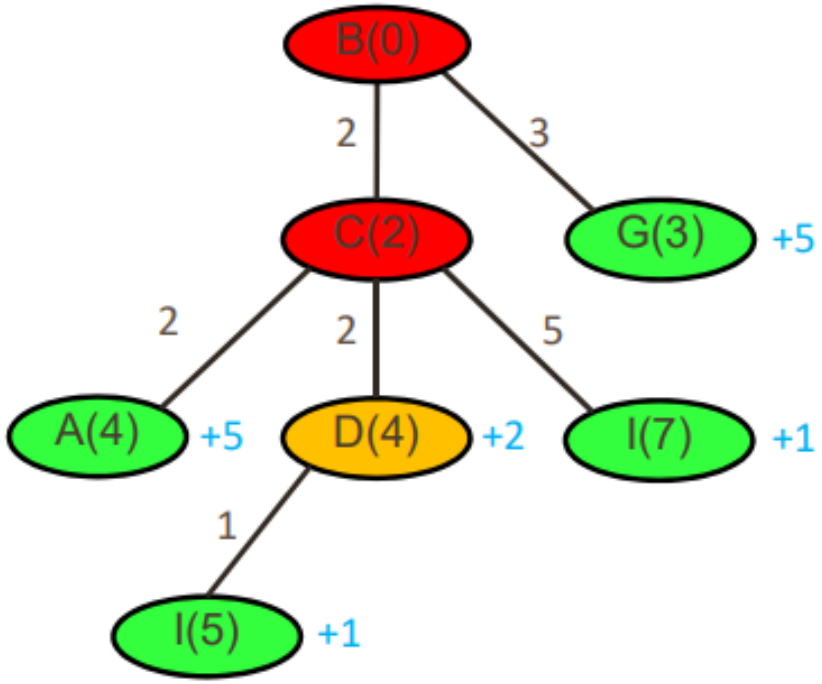


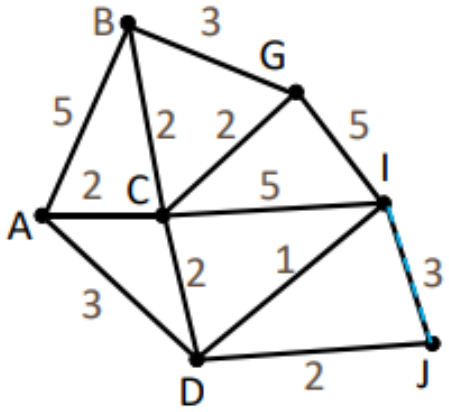
Open (Q):
{ I (5+1),
G (3+5),
A (4+5) }

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

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

A* heuristička pretraga



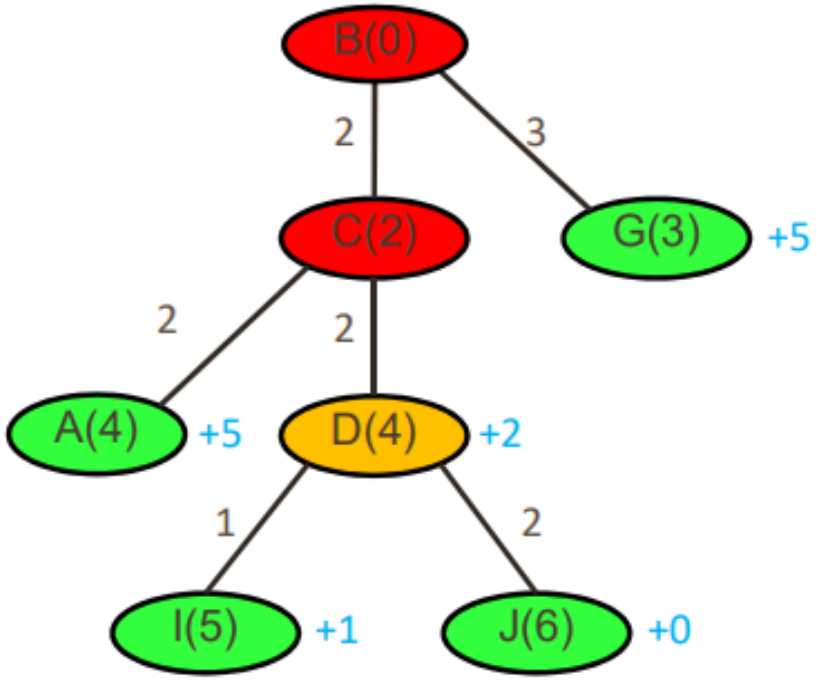


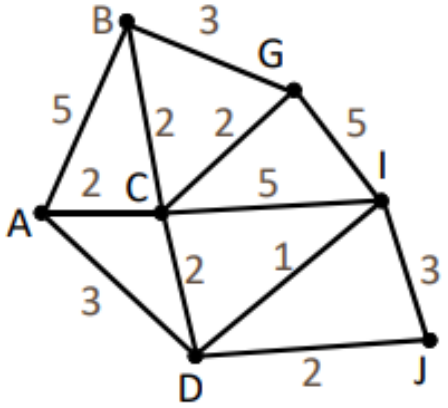
Open (Q):
{ I (5+1),
G (3+5),
A (4+5) }

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

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

A* heuristička pretraga



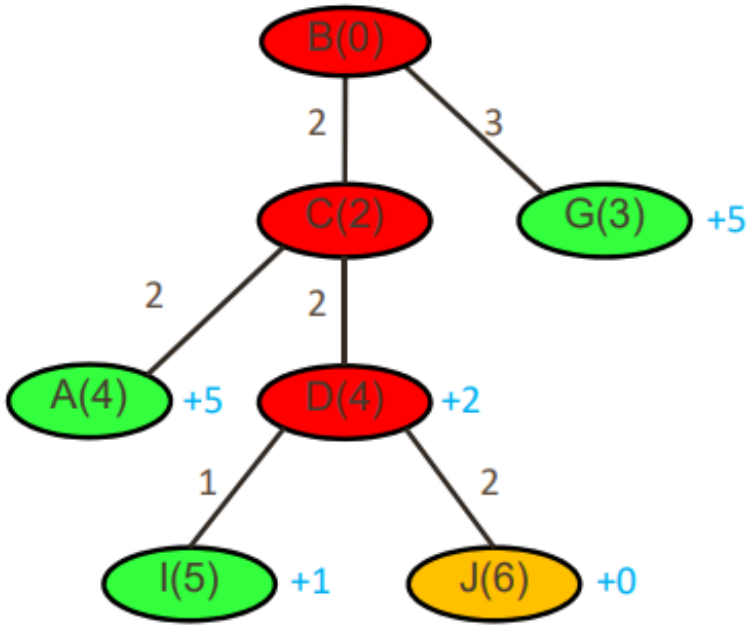


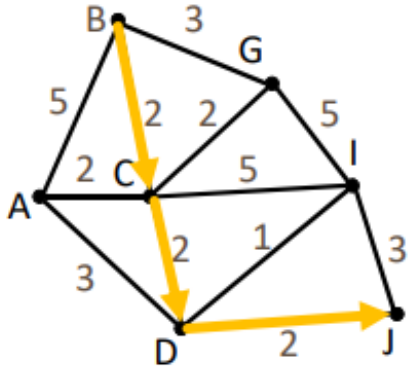
Open (Q):
{ J (6+0),
I (5+1),
G (3+5),
A (4+5) }

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

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

A* heuristička pretraga





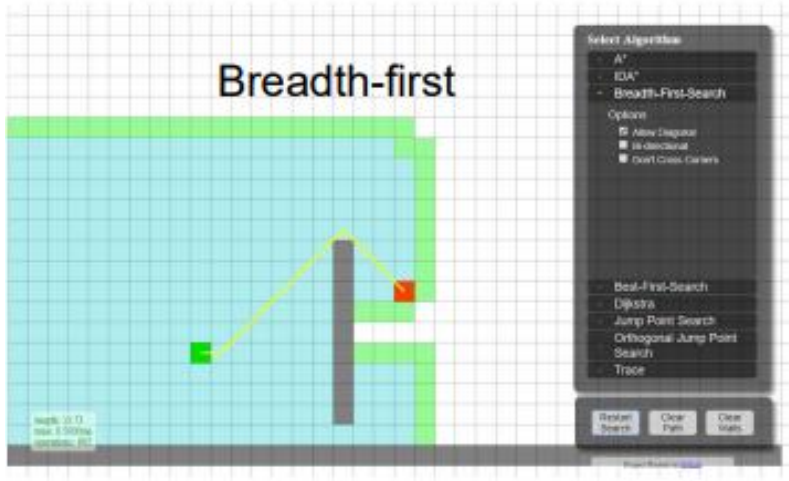
Open (Q):
{ I (5+1),
G (3+5),
A (4+5) }

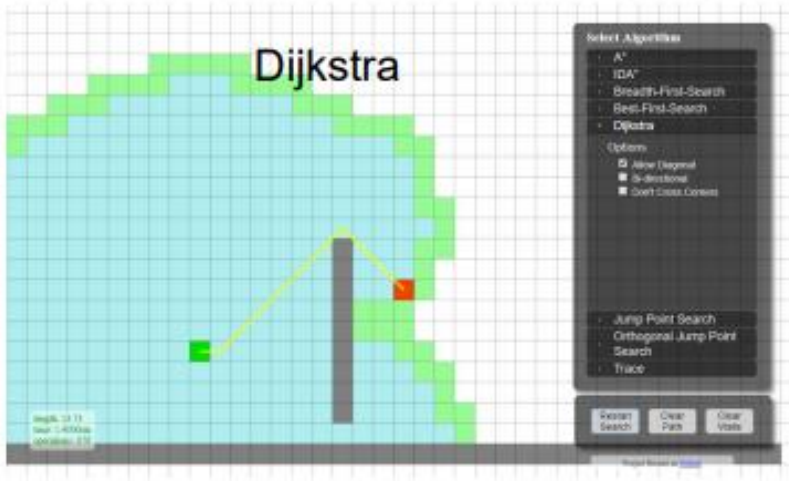
Closed:
{ B (0) ,
C (2),
D (4),
J (6) }

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