ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΑΘΑΝΑΣΙΟΥ ΙΩΑΝΝΗΣ

ΑΜ: 03117041

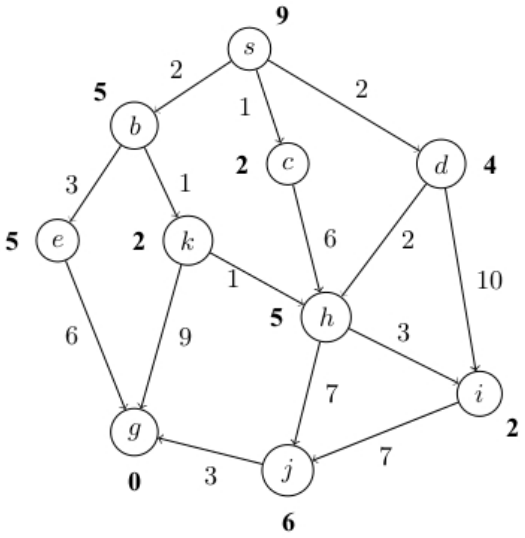
ΕΞΑΜΗΝΟ: 9ο

ΗΜΕΡΟΜΗΝΙΑ: 3/11/2021

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ - ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2021 - ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ 1

ΑΣΚΗΣΗ 1

- Για την άσκηση αυτή, έγραψα κώδικα σε Python για εξάσκηση και για να επαληθεύσω τα αποτελέσματα στα οποία κατέληξα εκτελώντας τον αλγόριθμο με το χέρι.
- Ο κώδικας αυτός βρίσκεται στο github repo (άφησα το repo να είναι public για να μπορείτε να δείτε τον κώδικα και εφόσον αυτός δεν αποτελεί ζητούμενο της άσκησης): https://github.com/John-Atha/AI-NTUA-21-22-writting-implementations



1. Hill climbing algorithm

- Αποτελέσματα εκτέλεσης Python:

```
atha@~/Desktop/AI/writting/week1/search:$ python3 search.py input.txt
Which algorithm would you like to use (1, 2 or 3)?
  1.Hill Climbing
  2.Best First
  3.A*
1
I am hill climbing, in fact I do not use a frontier, so you will not find a frontier on my table:
-------------------------------------------------
Current: s(9)
Closed set: {'s'}
Children: ['d(4)', 'b(5)', 'c(2)']
I am going to c(2)
-------------------------------------------------
Current: c(2)
Closed set: {'s', 'c'}
Children: ['h(5)']
No better children, no next state from c(2)
-------------------------------------------------
Could not reach goal, local minimum at: c(2)
With total cost:  1.0
atha@~/Desktop/AI/writting/week1/search:$
```

- Εκτέλεση με το χέρι:

current state	closed set	children
s(9)	s(9)	c(2), b(5), d(4)
c(2)	s(9), c(2)	h(5)

- Όπως φαίνεται και από την εκτέλεση του αλγορίθμου, το c(2) αποτελεί ένα τοπικό ελάχιστο της συνάρτησης κόστους, αφού ο κόμβος c(2), έχει

ως μοναδικό κόμβο “παιδί” του τον κόμβο h(5). Επομένως, ο αλγόριθμος hill-climbing δεν θα προχωρήσει σε επόμενη κατάσταση και θα τερματίσει χωρίς να βρει την λύση ‘g’.

2. Best-first algorithm

- Αποτελέσματα εκτέλεσης Python:

```
atha@~/Desktop/AI/writting/week1/search:$ python3 search.py input.txt
Which algorithm would you like to use (1, 2 or 3)?
  1.Hill Climbing
  2.Best First
  3.A*
2
----------------------------------------
Current: s(9)
Closed set: ['s(9)']
Children: ['b(5)', 'c(2)', 'd(4)']
Sorted frontier:
        b(5.0)(<-s(9)),
        d(4.0)(<-s(9)),
        c(2.0)(<-s(9)),

----------------------------------------
Current: c(2.0)(<-s(9)),
Closed set: ['c(2.0)<-s(9)', 's(9)']
Children: ['h(5)']
Sorted frontier:
        h(5.0)(<-c(2.0)<-s(9)),
        b(5.0)(<-s(9)),
        d(4.0)(<-s(9)),

----------------------------------------
Current: d(4.0)(<-s(9)),
Closed set: ['d(4.0)<-s(9)', 'c(2.0)<-s(9)', 's(9)']
Children: ['i(2)', 'h(5)']
Sorted frontier:
        h(5.0)(<-d(4.0)<-s(9)),
        h(5.0)(<-c(2.0)<-s(9)),
        b(5.0)(<-s(9)),
        i(2.0)(<-d(4.0)<-s(9)),

----------------------------------------
Current: i(2.0)(<-d(4.0)<-s(9)),
Closed set: ['d(4.0)<-s(9)', 'c(2.0)<-s(9)', 'i(2.0)<-d(4.0)', 's(9)']
Children: ['j(6)']
Sorted frontier:
        j(6.0)(<-i(2.0)<-d(4.0)<-s(9)),
        h(5.0)(<-d(4.0)<-s(9)),
        h(5.0)(<-c(2.0)<-s(9)),
        b(5.0)(<-s(9)),

----------------------------------------
Current: b(5.0)(<-s(9)),
Closed set: ['s(9)', 'd(4.0)<-s(9)', 'b(5.0)<-s(9)', 'i(2.0)<-d(4.0)', 'c(2.0)<-s(9)']
Children: ['e(5)', 'k(2)']
Sorted frontier:
        j(6.0)(<-i(2.0)<-d(4.0)<-s(9)),
        e(5.0)(<-b(5.0)<-s(9)),
        h(5.0)(<-d(4.0)<-s(9)),
        h(5.0)(<-c(2.0)<-s(9)),
        k(2.0)(<-b(5.0)<-s(9)),

----------------------------------------
Current: k(2.0)(<-b(5.0)<-s(9)),
Closed set: ['s(9)', 'k(2.0)<-b(5.0)', 'd(4.0)<-s(9)', 'b(5.0)<-s(9)', 'i(2.0)<-d(4.0)', 'c(2.0)<-s(9)']
Children: ['h(5)', 'g(0)']
Sorted frontier:
        j(6.0)(<-i(2.0)<-d(4.0)<-s(9)),
        h(5.0)(<-k(2.0)<-b(5.0)<-s(9)),
        e(5.0)(<-b(5.0)<-s(9)),
        h(5.0)(<-d(4.0)<-s(9)),
        h(5.0)(<-c(2.0)<-s(9)),
        g(0.0)(<-k(2.0)<-b(5.0)<-s(9)),

----------------------------------------
Found goal 'g', with path:
g(0.0)(<-k(2.0)<-b(5.0)<-s(9)), Total cost: 12.0
atha@~/Desktop/AI/writting/week1/search:$
```

- Εκτέλεση με το χέρι:

current state	**closed set**	**children**	**sorted frontier**
s(9)	(s, 9)	(b, 5)$^{b<-s}$ (d, 4)$^{d<-s}$ (c, 2)$^{c<-s}$	(b, 5)$^{b<-s}$ (d, 4)$^{d<-s}$ (c, 2)$^{c<-s}$
(c, 2)$^{c<-s}$	(s, 9) (c, 2)$^{c<-s}$	(h, 5) $^{h<-c<-s}$	(h, 5) $^{h<-c<-s}$ (b, 5)$^{b<-s}$ (d, 4)$^{d<-s}$
(d, 4)$^{d<-s}$	(s, 9) (c, 2)$^{c<-s}$ (d, 4)$^{d<-s}$	(h, 5)$^{h<-d<-s}$ (i, 2)$^{i<-d<-s}$	(h, 5)$^{h<-d<-s}$ (h, 5)$^{h<-c<-s}$ (b, 5)$^{b<-s}$ (i, 2)$^{i<-d<-s}$
(i, 2)$^{i<-d<-s}$	(s, 9) (c, 2)$^{c<-s}$ (d, 4)$^{d<-s}$ (i, 2)$^{i<-d<-s}$	(j, 6)$^{j<-i<-d<-s}$	(j, 6)$^{j<-i<-d<-s}$ (h, 5)$^{h<-d<-s}$ (h, 5)$^{h<-c<-s}$ (b, 5)$^{b<-s}$
(b, 5)$^{b<-s}$	(s, 9) (c, 2)$^{c<-s}$ (d, 4)$^{d<-s}$ (i, 2)$^{i<-d<-s}$ (b, 5)$^{b<-s}$	(e, 5)$^{e<-b<-s}$ (k, 2)$^{k<-b<-s}$	(j, 6)$^{j<-i<-d<-s}$ (e, 5)$^{e<-b<-s}$ (h, 5)$^{h<-d<-s}$ (h, 5)$^{h<-c<-s}$ (k, 2)$^{k<-b<-s}$
(k, 2)$^{k<-b<-s}$	(s, 9) (c, 2)$^{c<-s}$ (d, 4)$^{d<-s}$ (i, 2)$^{i<-d<-s}$ (b, 5)$^{b<-s}$ (k, 2)$^{k<-b<-s}$	(g, 0)$^{g<-k<-b<-s}$ (h, 5)$^{h<-k<-b<-s}$	(j, 6)$^{j<-i<-d<-s}$ (h, 5)$^{h<-k<-b<-s}$ (e, 5)$^{e<-b<-s}$ (h, 5)$^{h<-d<-s}$ (h, 5)$^{h<-c<-s}$ (g, 0)$^{g<-k<-b<-s}$
(g, 0)$^{g<-k<-b<-s}$			

- Ο αλγόριθμος τερματίζει επιτυχώς, αλλά παρατηρώ ότι δεν βρίσκει το συντομότερο μονοπάτι (βρίσκει μονοπάτι με κόστος 12, ενώ παρατηρώ ότι υπάρχει και συντομότερο μονοπάτι με κόστος 11).

3. A algorithm*

- *Θεωρώ ότι μία κατάσταση εισάγεται στο κλειστό σύνολο συνοδευόμενη από το μονοπάτι της.*
- *Δηλαδή εάν στο κλειστό σύνολο έχω τον κόμβο ‘u’ μέσω του μονοπατιού ‘p1’, και στην συνέχεια εξάγω από το μέτωπο αναζήτησης τον κόμβο ‘u’ μέσω του μονοπατιού ‘p2’, τότε ο κόμβος ‘u’ θα επεκταθεί και πάλι.*

- Αποτελέσματα εκτέλεσης Python:

```
atha@~/Desktop/AI/writting/week1/search:$ python3 search.py input.txt
Which algorithm would you like to use (1, 2 or 3)?
  1.Hill Climbing
  2.Best First
  3.A*
3
----------------------------------------
Current: s(9)
Closed set: ['s(9)']
Children: ['c(2)', 'd(4)', 'b(5)']
Sorted frontier:
        b(5.0, 2.0)(<-2.0<-s(9)),
        d(4.0, 2.0)(<-2.0<-s(9)),
        c(2.0, 1.0)(<-1.0<-s(9)),

----------------------------------------
Current: c(2.0, 1.0)(<-s(9)),
Closed set: ['c(2.0)<-s(9)', 's(9)']
Children: ['h(5)']
Sorted frontier:
        h(5.0, 7.0)(<-6.0<-c(2.0)<-1.0-s(9)),
        b(5.0, 2.0)(<-2.0<-s(9)),
        d(4.0, 2.0)(<-2.0<-s(9)),

----------------------------------------
Current: d(4.0, 2.0)(<-s(9)),
Closed set: ['d(4.0)<-s(9)', 'c(2.0)<-s(9)', 's(9)']
Children: ['h(5)', 'i(2)']
Sorted frontier:
        i(2.0, 12.0)(<-10.0<-d(4.0)<-2.0-s(9)),
        h(5.0, 4.0)(<-2.0<-d(4.0)<-2.0-s(9)),
        b(5.0, 2.0)(<-2.0<-s(9)),

----------------------------------------
Current: b(5.0, 2.0)(<-s(9)),
Closed set: ['d(4.0)<-s(9)', 'c(2.0)<-s(9)', 'b(5.0)<-s(9)', 's(9)']
Children: ['e(5)', 'k(2)']
Sorted frontier:
        i(2.0, 12.0)(<-10.0<-d(4.0)<-2.0-s(9)),
        e(5.0, 5.0)(<-3.0<-b(5.0)<-2.0-s(9)),
        h(5.0, 4.0)(<-2.0<-d(4.0)<-2.0-s(9)),
        k(2.0, 3.0)(<-1.0<-b(5.0)<-2.0-s(9)),

----------------------------------------
Current: k(2.0, 3.0)(<-b(5.0)<-s(9)),
Closed set: ['b(5.0)<-s(9)', 'c(2.0)<-s(9)', 'd(4.0)<-s(9)', 's(9)', 'k(2.0)<-b(5.0)']
Children: ['h(5)', 'g(0)']
Sorted frontier:
        i(2.0, 12.0)(<-10.0<-d(4.0)<-2.0-s(9)),
        g(0.0, 12.0)(<-9.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        e(5.0, 5.0)(<-3.0<-b(5.0)<-2.0-s(9)),
        h(5.0, 4.0)(<-1.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        h(5.0, 4.0)(<-2.0<-d(4.0)<-2.0-s(9)),

----------------------------------------
Current: h(5.0, 4.0)(<-d(4.0)<-s(9)),
Closed set: ['b(5.0)<-s(9)', 'c(2.0)<-s(9)', 'h(5.0)<-d(4.0)', 'd(4.0)<-s(9)', 's(9)', 'k(2.0)<-b(5.0)']
Children: ['i(2)', 'j(6)']
Sorted frontier:
        j(6.0, 11.0)(<-7.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),
        g(0.0, 12.0)(<-9.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        e(5.0, 5.0)(<-3.0<-b(5.0)<-2.0-s(9)),
        i(2.0, 7.0)(<-3.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),
        h(5.0, 4.0)(<-1.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),

----------------------------------------
Current: h(5.0, 4.0)(<-k(2.0)<-b(5.0)<-s(9)),
Closed set: ['b(5.0)<-s(9)', 'h(5.0)<-k(2.0)', 'c(2.0)<-s(9)', 'h(5.0)<-d(4.0)', 'd(4.0)<-s(9)', 's(9)', 'k(2.0)<-b(5.0)']
Children: ['i(2)', 'j(6)']
Sorted frontier:
        j(6.0, 11.0)(<-7.0<-h(5.0)<-1.0-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        j(6.0, 11.0)(<-7.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),
        g(0.0, 12.0)(<-9.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        e(5.0, 5.0)(<-3.0<-b(5.0)<-2.0-s(9)),
        i(2.0, 7.0)(<-3.0<-h(5.0)<-1.0-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        i(2.0, 7.0)(<-3.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),

----------------------------------------
Current: i(2.0, 7.0)(<-h(5.0)<-d(4.0)<-s(9)),
Closed set: ['i(2.0)<-h(5.0)', 'b(5.0)<-s(9)', 'h(5.0)<-k(2.0)', 'c(2.0)<-s(9)', 'h(5.0)<-d(4.0)', 'd(4.0)<-s(9)', 's(9)', 'k(2.0)<-b(5.0)']
Children: ['j(6)']
Sorted frontier:
        j(6.0, 11.0)(<-7.0<-h(5.0)<-1.0-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        j(6.0, 11.0)(<-7.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),
        g(0.0, 12.0)(<-9.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        e(5.0, 5.0)(<-3.0<-b(5.0)<-2.0-s(9)),
        i(2.0, 7.0)(<-3.0<-h(5.0)<-1.0-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),

----------------------------------------
Current: i(2.0, 7.0)(<-h(5.0)<-k(2.0)<-b(5.0)<-s(9)),
Closed set: ['i(2.0)<-h(5.0)', 'b(5.0)<-s(9)', 'h(5.0)<-k(2.0)', 'c(2.0)<-s(9)', 'h(5.0)<-d(4.0)', 'd(4.0)<-s(9)', 's(9)', 'k(2.0)<-b(5.0)', 'i(2.0)<-h(5.0)']
Children: ['j(6)']
Sorted frontier:
        j(6.0, 11.0)(<-7.0<-h(5.0)<-1.0-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        j(6.0, 11.0)(<-7.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),
        g(0.0, 12.0)(<-9.0<-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        e(5.0, 5.0)(<-3.0<-b(5.0)<-2.0-s(9)),

----------------------------------------
Current: e(5.0, 5.0)(<-b(5.0)<-s(9)),
Closed set: ['i(2.0)<-h(5.0)', 'b(5.0)<-s(9)', 'h(5.0)<-k(2.0)', 'e(5.0)<-b(5.0)', 'c(2.0)<-s(9)', 'h(5.0)<-d(4.0)', 'd(4.0)<-s(9)', 's(9)', 'k(2.0)<-b(5.0)', 'i(2.0)<-h(5.0)']
Children: ['g(0)']
Sorted frontier:
        j(6.0, 11.0)(<-7.0<-h(5.0)<-1.0-k(2.0)<-1.0-b(5.0)<-2.0-s(9)),
        j(6.0, 11.0)(<-7.0<-h(5.0)<-2.0-d(4.0)<-2.0-s(9)),
        g(0.0, 11.0)(<-6.0<-e(5.0)<-3.0-b(5.0)<-2.0-s(9)),

----------------------------------------
Found goal 'g', with path:
g(0.0, 11.0)(<-e(5.0)<-b(5.0)<-s(9)), Total cost: 11.0
atha@~/Desktop/AI/writting/week1/search:$
```

- Εκτέλεση με το χέρι:

current state	closed set	children	sorted frontier
(s, 9, 0)	(s, 9, 0)	(b, 5, 2)$^{b<-s}$ (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$	(b, 5, 2)$^{b<-s}$ (d, 4, 2)$^{d<-s}$ (c, 2, 1)$^{c<-s}$
(c, 2, 1)$^{c<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$	(h, 5, 7)$^{h<-c<-s}$	(h, 5, 7)$^{h<-c<-s}$ (b, 5, 2)$^{b<-s}$ (d, 4, 2)$^{d<-s}$
(d, 4, 2)$^{d<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$	(h, 5, 4)$^{h<-d<-s}$ (i, 2, 12)$^{i<-d<-s}$	(i, 2, 12)$^{i<-d<-s}$ (h, 5, 4)$^{h<-d<-s}$ (b, 5, 2)$^{b<-s}$
(b, 5, 2)$^{b<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$	(e, 5, 5)$^{e<-b<-s}$ (k, 2, 3)$^{k<-b<-s}$	(i, 2, 12)$^{i<-d<-s}$ (e, 5, 5)$^{e<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$ (k, 2, 3)$^{k<-b<-s}$
(k, 2, 3)$^{k<-b<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$ (k, 2, 3)$^{k<-b<-s}$	(g, 0, 12)$^{g<-k<-b<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$	(i, 2, 12)$^{i<-d<-s}$ (g, 0, 12)$^{g<-k<-b<-s}$ (e, 5, 5)$^{e<-b<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$
(h, 5, 4)$^{h<-d<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$ (k, 2, 3)$^{k<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$	(i, 2, 7)$^{i<-h<-d<-s}$ (j, 6, 11)$^{j<-h<-d<-s}$	(j, 6, 11)$^{j<-h<-d<-s}$ (g, 0, 12)$^{g<-k<-b<-s}$ (e, 5, 5)$^{e<-b<-s}$ (i, 2, 7)$^{i<-h<-d<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$
(h, 5, 4)$^{h<-k<-b<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$ (k, 2, 3)$^{k<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$	(i, 2, 7)$^{i<-h<-k<-b<-s}$ (j, 6, 11)$^{j<-h<-k<-b<-s}$	(j, 6, 11)$^{j<-h<-k<-b<-s}$ (j, 6, 11)$^{j<-h<-d<-s}$ (g, 0, 12)$^{g<-k<-b<-s}$ (e, 5, 5)$^{e<-b<-s}$ (i, 2, 7)$^{i<-h<-k<-b<-s}$ (i, 2, 7)$^{i<-h<-d<-s}$
(i, 2, 7)$^{i<-h<-d<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$ (k, 2, 3)$^{k<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$ (i, 2, 7)$^{i<-h<-d<-s}$	(j, 6, 14)$^{j<-i<-h<-d<-s}$	(j, 6, 11)$^{j<-h<-k<-b<-s}$ (j, 6, 11)$^{j<-h<-d<-s}$ (g, 0, 12)$^{g<-k<-b<-s}$ (e, 5, 5)$^{e<-b<-s}$ (i, 2, 7)$^{i<-h<-k<-b<-s}$
(i, 2, 7)$^{i<-h<-k<-b<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$ (k, 2, 3)$^{k<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$	(j, 6, 14)$^{j<-i<-h<-k<-b<-s}$	(j, 6, 11)$^{j<-h<-k<-b<-s}$ (j, 6, 11)$^{j<-h<-d<-s}$ (g, 0, 12)$^{g<-k<-b<-s}$ (e, 5, 5)$^{e<-b<-s}$

	(i, 2, 7)$^{i<-h<-d<-s}$ (i, 2, 7)$^{i<-h<-k<-b<-s}$		
(e, 5, 5)$^{e<-b<-s}$	(s, 9, 0) (c, 2, 1)$^{c<-s}$ (d, 4, 2)$^{d<-s}$ (b, 5, 2)$^{b<-s}$ (k, 2, 3)$^{k<-b<-s}$ (h, 5, 4)$^{h<-d<-s}$ (h, 5, 4)$^{h<-k<-b<-s}$ (i, 2, 7)$^{i<-h<-d<-s}$ (i, 2, 7)$^{i<-h<-k<-b<-s}$ (e, 5, 5)$^{e<-b<-s}$	(g, 0, 11)$^{g<-e<-b<-s}$	(j, 6, 11)$^{j<-h<-k<-b<-s}$ (j, 6, 11)$^{j<-h<-d<-s}$ (g, 0, 11)$^{g<-e<-b<-s}$
(g, 0, 11)$^{g<-e<-b<-s}$			

- Παρατηρώ δύο διαδοχικές επεκτάσεις των κόμβων ‘i’ και ‘h’ καθώς κάθε φορά προέρχονταν από διαφορετικά μονοπάτια.
- Παρατηρώ ότι ο αλγόριθμος τερματίζει επιτυχώς, βρίσκοντας μονοπάτι μήκους 11 (στο συγκεριμένο παράδειγμα αυτό φαίνεται να είναι το συντομότερο μονοπάτι), άρα στο συγκεκριμένο τουλάχιστον παράδειγμα είναι βέλτιστος.

2. Για να βρούμε όλα τα μονοπάτια από τον κόμβο ‘s’ στον κόμβο ‘g’, μπορούμε να εκτελέσουμε DFS, κρατώντας τον πατέρα κάθε κόμβου και κάθε φορά που βρίσκουμε ένα νεό μονοπάτι να το κρατάμε.

Εκτελώντας DFS με το χέρι μετράω 9 μονοπάτια (9 λύσεις του προβλήματος) και για επαλήθευση εκτελώ και αντίστοιχο τμήμα κώδικα Python, κι έχω την έξοδο:

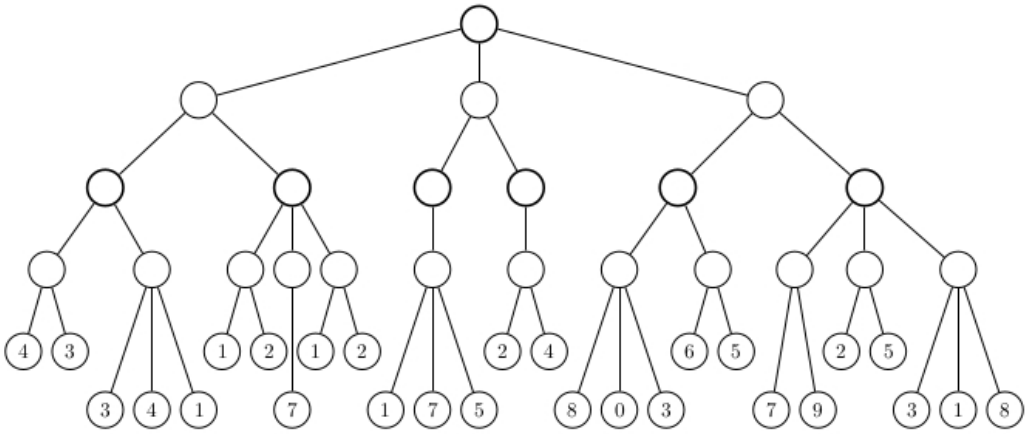
```
atha@~/Desktop/AI/writting/week1/search:$ python3 search.py input.txt
Which algorithm would you like to use (1, 2, 3 or 4)?
  1.Hill Climbing
  2.Best First
  3.A*
  4.DFS to count all paths from 's' to 'g'
4
Found goal
Found goal
Found goal
Found goal
Found goal
Found goal
Found goal
Found goal
Found goal
DFS is over
Total paths found: 9
g <- j <- i <- h <- c <- s with cost: 20.0
g <- j <- i <- h <- k <- b <- s with cost: 17.0
g <- j <- i <- d <- s with cost: 22.0
g <- j <- h <- c <- s with cost: 17.0
g <- j <- h <- d <- s with cost: 14.0
g <- j <- h <- k <- b <- s with cost: 14.0
g <- k <- b <- s with cost: 12.0
g <- j <- i <- h <- d <- s with cost: 17.0
g <- e <- b <- s with cost: 11.0
atha@~/Desktop/AI/writting/week1/search:$
```

- Όπως έχω αναφέρει ήδη, η βέλτιστη λύση βρέθηκε από τον αλγόριθμο A* και είναι το τελευταίο μονοπάτι της εξόδου.

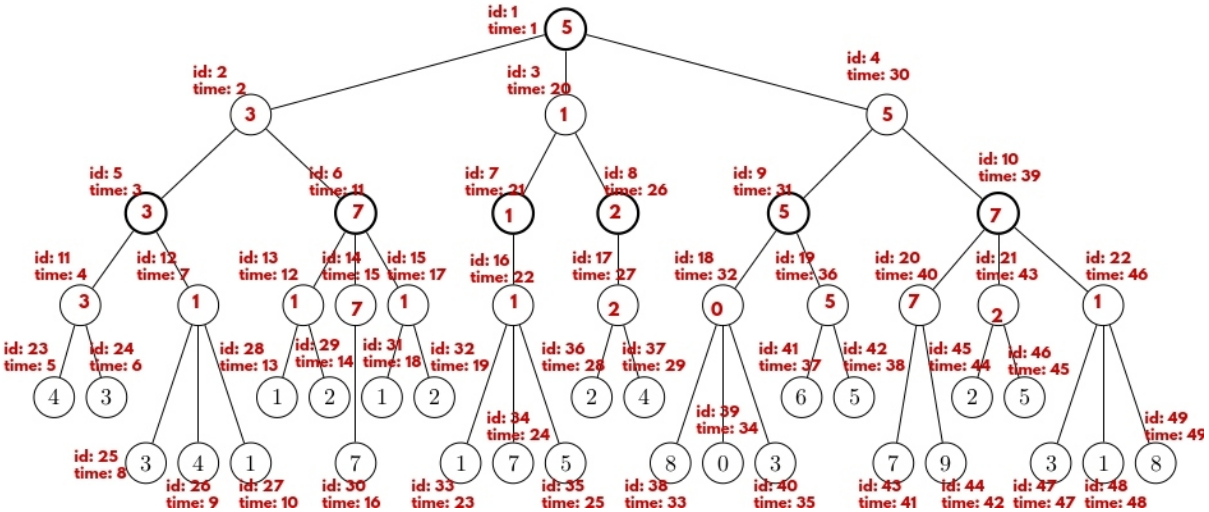
- Γενικά ο αλγόριθμος A* βρίσκει την βέλτιστη λύση όταν ο ευρετικός μηχανισμός είναι αποδεκτός (admissible) δηλαδή η τιμή της ευρετικής είναι μικρότερη ή ίση με την πραγματική τιμή.
- Στο συγκεκριμένο παράδειγμα, παρατηρώ ότι αυτό δεν συμβαίνει σε όλους τους κόμβους, αφού, για παράδειγμα, ο κόμβος ‘j’ έχει ευρετική τιμή 6 και πραγματική τιμή 3.
- Επομένως, δεν θα μπορούσαμε εκ των προτέρων να είμαστε σίγουροι ότι ο A* θα έβρισκε βέλτιστη λύση.

ΑΣΚΗΣΗ 2

- Για την άσκηση αυτή, έγραψα κώδικα σε Python για εξάσκηση και για να επαληθεύσω τα αποτελέσματα στα οποία κατέληξα εκτελώντας τον αλγόριθμο με το χέρι.
- Ο κώδικας αυτός βρίσκεται στο github repo (άφησα το repo να είναι public για να μπορείτε να δείτε τον κώδικα και εφόσον αυτός δεν αποτελεί ζητούμενο της άσκησης): https://github.com/John-Atha/AI-NTUA-21-22-writting-implementations



1. Εκτελώ τον αλγόριθμο με το χέρι, διαλέγοντας την μέγιστη τιμή των παιδιών για τους maximizing κόμβους, και την ελάχιστη τιμή των παιδιών για τους minimizing κόμβους, κι έχω τα εξής αποτελέσματα:



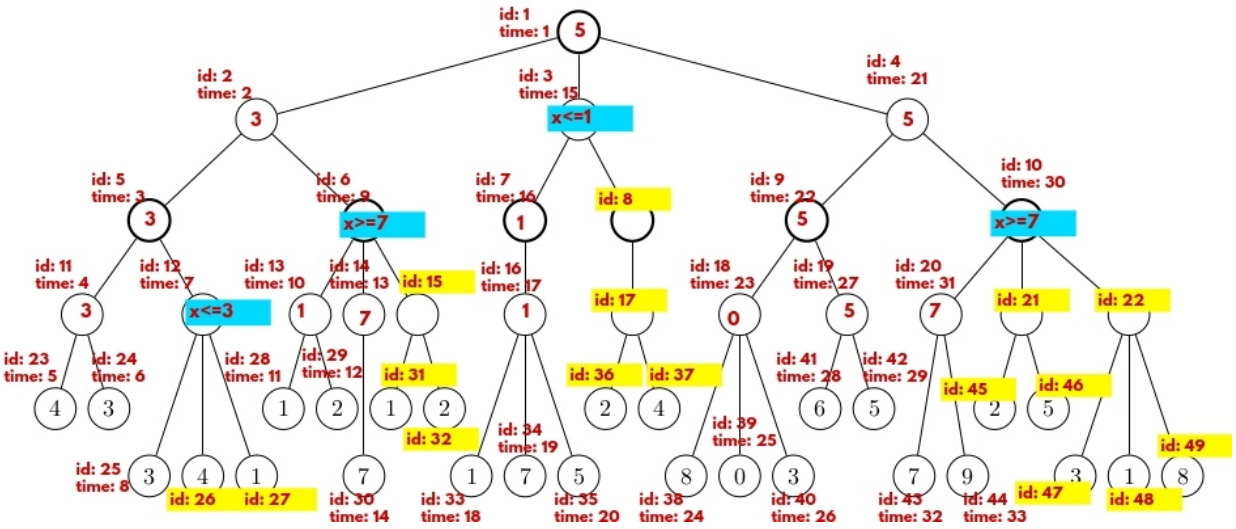
- Σε κάθε κόμβο έχω σημειώσει το id του, την σειρά (time) με την οποία επισκέφθηκε από τον αλγόριθμο και την τιμή που παίρνει μετά την εκτέλεση.

- Εκτελώ για το ίδιο παράδειγμα το πρόγραμμα Python κι έχω την έξοδο:

```
atha@~/Desktop/AI/writting/week1/minimax:$ python3 minimax.py input.txt
I am minimax without AB pruning
 * Root value: 5
 * Nodes visited counter: 50
 * Nodes visited:
        u1: 5->         u2: 3->         u5: 3->         u11: 3->        u23: 4->
        u24: 3->        u12: 1->        u25: 3->        u26: 4->        u27: 1->
        u6: 7->         u13: 1->        u28: 1->        u29: 2->        u14: 7->
        u30: 7->        u15: 1->        u31: 1->        u32: 2->        u3: 1->
        u7: 1->         u16: 1->        u33: 1->        u34: 7->        u35: 5->
        u8: 2->         u17: 2->        u36: 2->        u37: 4->        u4: 5->
        u9: 5->         u18: 0->        u38: 8->        u39: 0->        u40: 3->
        u19: 5->        u41: 6->        u42: 5->        u10: 7->        u20: 7->
        u43: 7->        u44: 9->        u21: 2->        u45: 2->        u46: 5->
        u22: 1->        u47: 3->        u48: 1->        u49: 8
  * Not visited nodes:
        []
I am minimax with AB pruning
 * Root value: 5
 * Nodes visited counter: 34
 * Nodes visited:
        u1: 5->         u2: 3->         u5: 3->         u11: 3->        u23: 4->
        u24: 3->        u12: None->     u25: 3->        u6: None->      u13: 1->
        u28: 1->        u29: 2->        u14: 7->        u30: 7->        u3: None->
        u7: 1->         u16: 1->        u33: 1->        u34: 7->        u35: 5->
        u4: 5->         u9: 5->         u18: 0->        u38: 8->        u39: 0->
        u40: 3->        u19: 5->        u41: 6->        u42: 5->        u10: None->
        u20: 7->        u43: 7->        u44: 9
  * Not visited nodes:
        [8, 15, 17, 21, 22, 26, 27, 31, 32, 36, 37, 45, 46, 47, 48, 49]
atha@~/Desktop/AI/writting/week1/minimax:$
```

- Προς το παρόν αγνοώ τα αποτελέσματα χρησιμοποιώντας AB pruning.
- Παρατηρώ ότι αναφορικά με το ερώτημα (1), τα αποτελέσματα είναι σε συμφωνία.

2. Εκτελώ τον αλγόριθμο minimax με την χρήση AB pruning και βρίσκω τα αποτελέσματα:



- Σε κάθε κόμβο σημειώνονται τα ίδια στοιχεία με προηγουμένως.
- Έχω τονίσει με κίτρινο background τους κόμβους τους οποίους ο αλγόριθμος δεν θα επισκεφθεί, καθώς η επίσκεψή τους κρίνεται περιττή εξαιτίας των σχέσεων στους κόμβους με μπλε background.
- Παρατηρώ ότι με την χρήση του AB pruning, “γλίτωσα” την επίσκεψη 16 κόμβων.
- Η σειρά επίσκεψης των κόμβων, όπως φαίνεται και από τους χρόνους που έχουν σημειωθεί στο σχήμα, είναι η εξής:
  1->2->5->11->23->24->12->25->6->13->28->29->14->30->3->7->16->
  33->34->35->4->9->18->38->39->20->43->44
- Οι κόμβοι που δεν επισκέφθηκαν από τον αλγόριθμο είναι οι:
  8, 15, 17, 21, 22, 26, 27, 31, 32, 36, 37, 45, 46, 47, 48, 49
- Τα αποτελέσματα της εκτέλεσης του αλγορίθμου με το χέρι είναι σε συμφωνία με τα αποτελέσματα της εκτέλεσης του προγράμματος Python, που παρατίθενται στο ερώτημα 1.