Διάλεξη 23: Γράφοι IV - Βραχύτερα Μονοπάτια σε Γράφους

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Βραχύτερα Μονοπάτια σε γράφους
- Ο αλγόριθμος Dijkstra για εύρεση της βραχύτερης απόστασης
- Ο αλγόριθμος Dijkstra για εύρεση του βραχύτερου μονοπατιού & απόστασης

Βραχύτερα Μονοπάτια σε Γράφους

- Με δεδομένο ένα **κατευθυνόμενο** γράφο με βάρη G=(V,E), θέλουμε να βρούμε τα μονοπάτια με το ελάχιστο δυνατό βάρος από κάποιο κόμβο Α προς οποιονδήποτε κόμβο Χ.
- **Ορισμός: *Βραχύτερο μονοπάτι*** μεταξύ ενός συνόλου από μονοπάτια είναι το μονοπάτι με το ελάχιστο βάρος.
- Υπενθύμιση: Το βάρος w(p) ενός μονοπατιού p δίνεται ως εξής:

$$\begin{aligned} &\text{Αν} \quad p = v_o \rightarrow v_1 \rightarrow \ldots \rightarrow v_k \\ &\text{τότε} \quad w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \end{aligned}$$

- Σε αυτή την διάλεξη θα δούμε ακόμα ένα άπληστο αλγόριθμο (greedy algorithm) του Δανού **Edsger Dijkstra** (1930-2002) για την επίλυση αυτού του προβλήματος.
- Ένας τέτοιος αλγόριθμος έχει πολλές εφαρμογές (π.χ. εύρεση μικρότερης διαδρομής σε ένα οδικό δίκτυο, σε δίκτυα υπολογιστών κτλ).

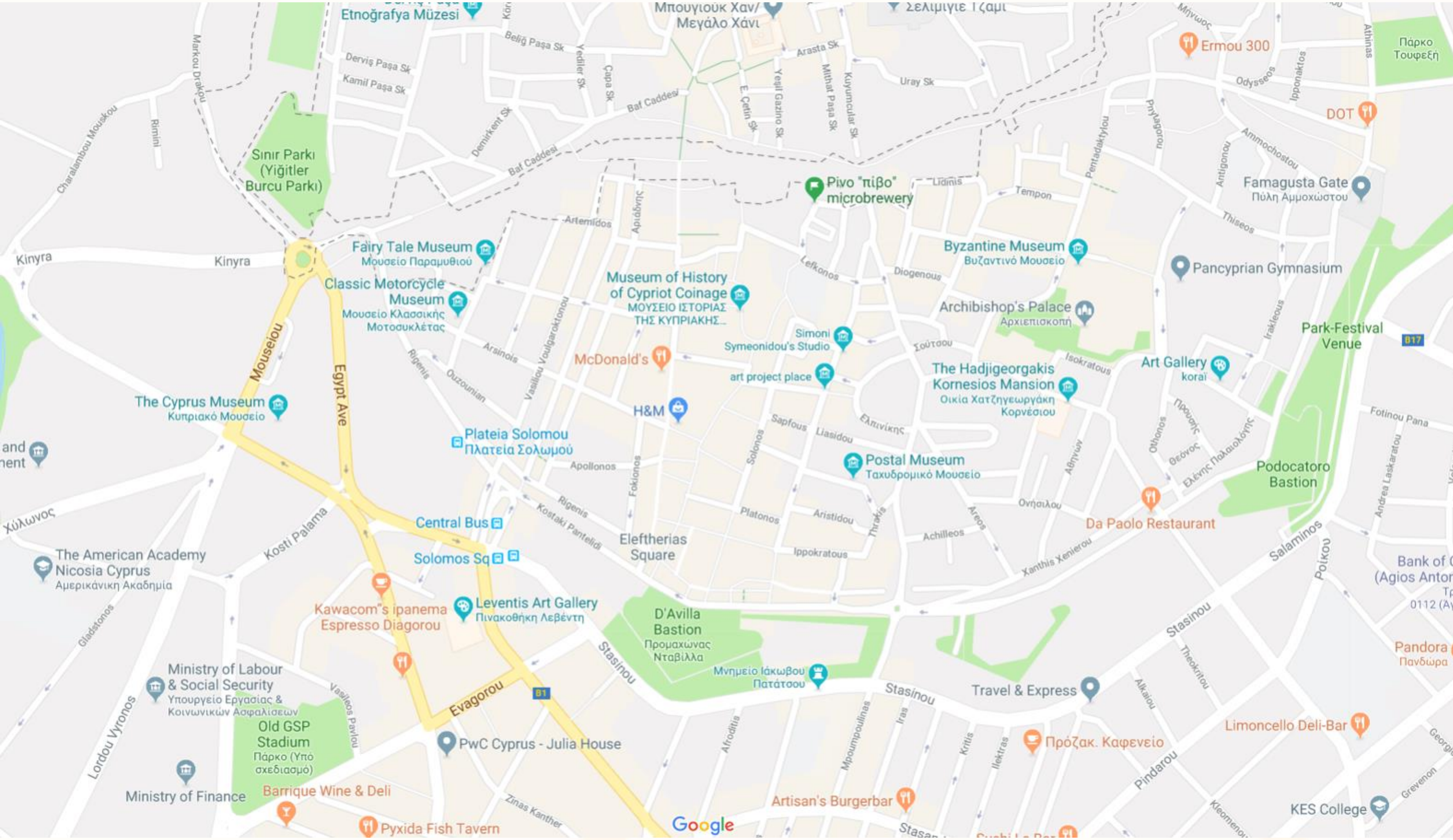
Shortest path applications

- PERT/CPM.
- Map routing.
- Seam carving.
- Robot navigation.
- Texture mapping.
- Typesetting in TeX.
- Urban traffic planning.
- Optimal pipelining of VLSI chip.
- Telemarketer operator scheduling.
- Routing of telecommunications messages.
- Network routing protocols (OSPF, BGP, RIP).
- Exploiting arbitrage opportunities in currency exchange.
- Optimal truck routing through given traffic congestion pattern.

http://en.wikipedia.org/wiki/Seam_carving

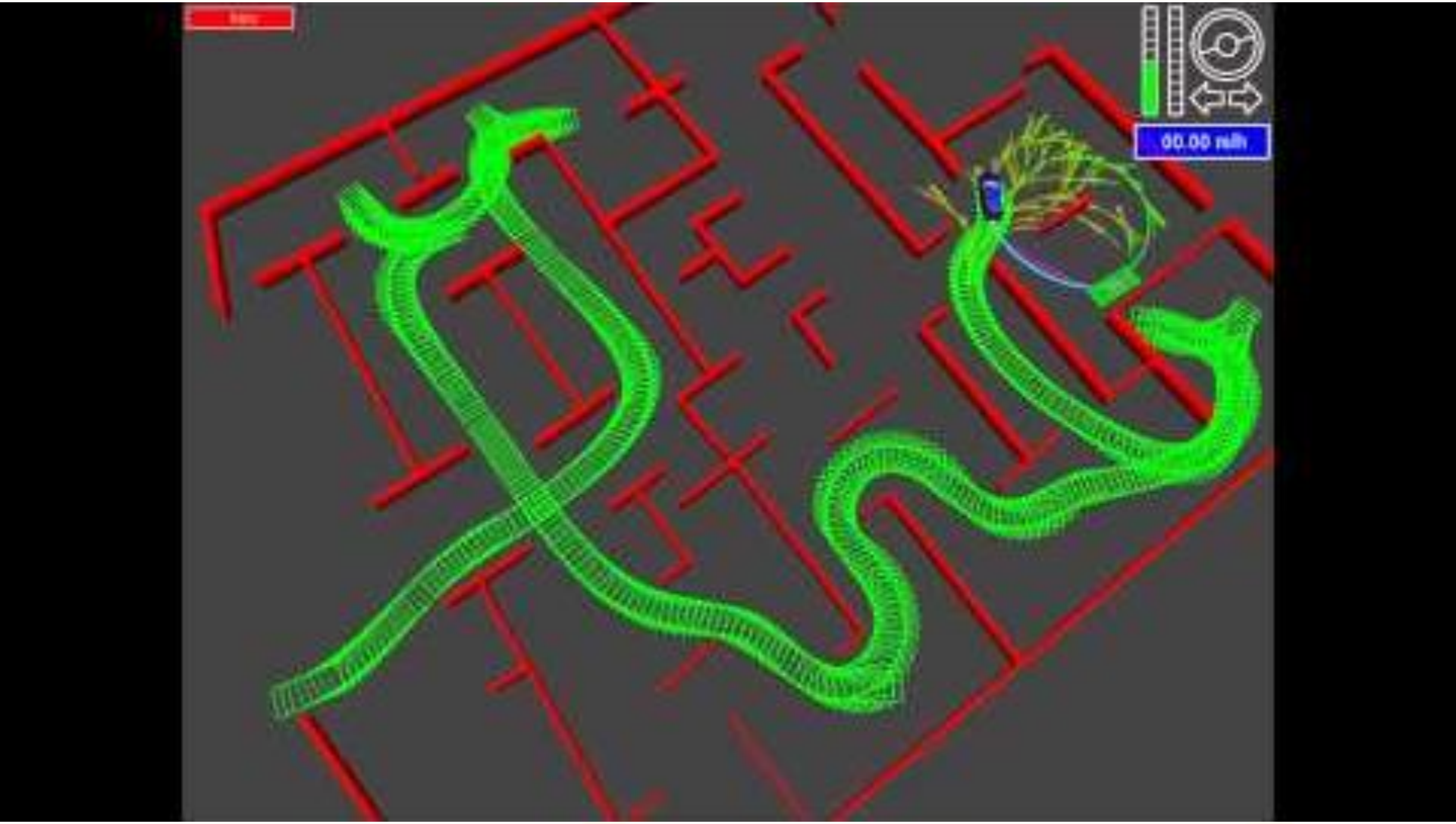
Reference: Network Flows: Theory, Algorithms, and Applications, R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Prentice Hall, 1993.

Map Routing



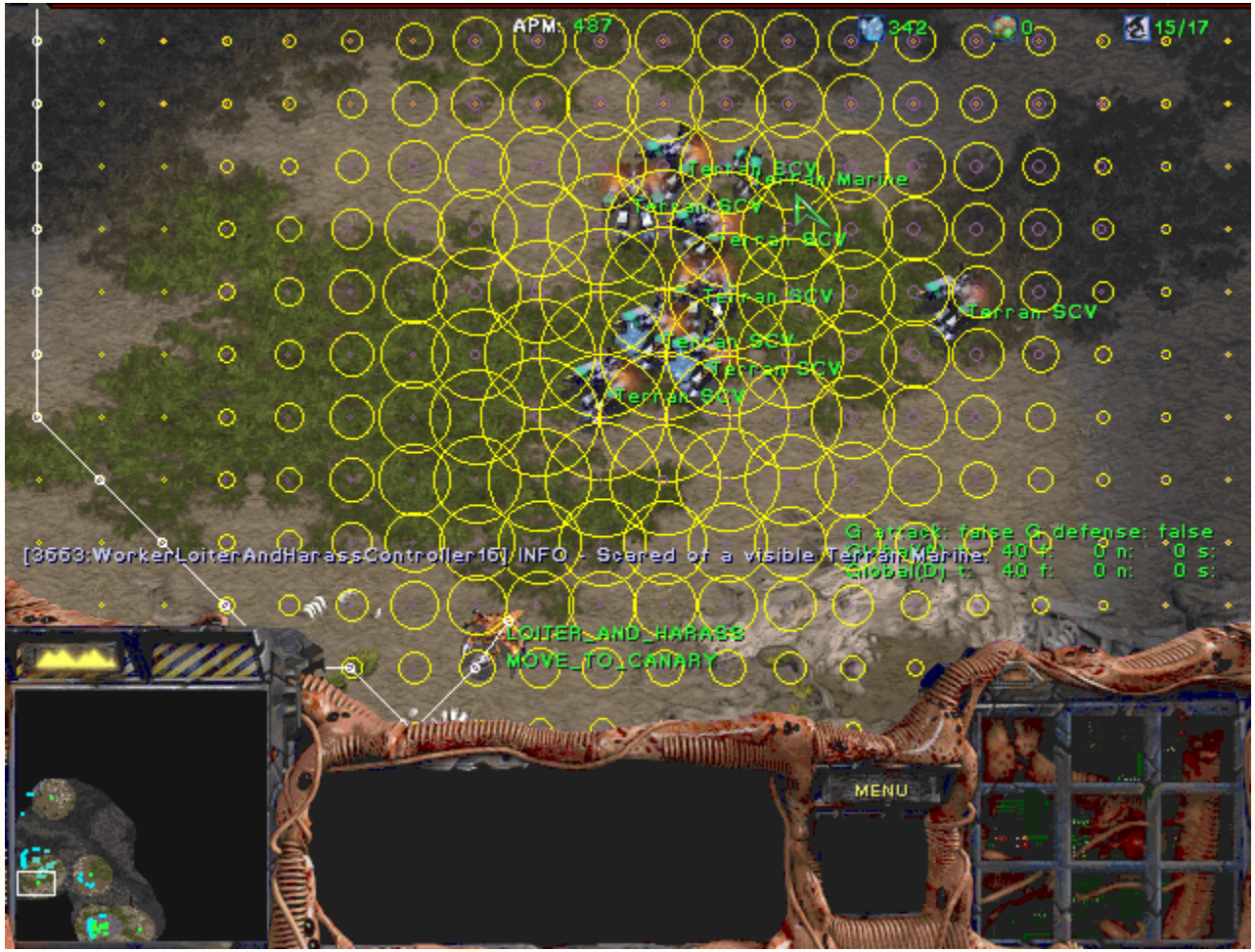
Etnoğrafya Müzesi
Μπουγιούκ Χαν/
Μεγάλο Χάνι
Ermou 300
Πάρκο
Τουφεξή
DOT
Sınır Parkı
(Yiğitler
Burcu Parkı)
Pivo "πίβο"
microbrewery
Famagusta Gate
Πύλη Αμμοχώστου
Kinyra
Fairy Tale Museum
Μουσείο Παραμυθιού
Classic Motorcycle
Museum
Μουσείο Κλασσικής
Μοτοσυκλέτας
Museum of History
of Cypriot Coinage
ΜΟΥΣΕΙΟ ΙΣΤΟΡΙΑΣ
ΤΗΣ ΚΥΠΡΙΑΚΗΣ...
Byzantine Museum
Βυζαντινό Μουσείο
Pancyprian Gymnasium
Archibishop's Palace
Αρχιεπισκοπή
Park-Festival
Venue
Simoni
Symeonidou's Studio
McDonald's
art project place
The Hadjigeorgakis
Kornesios Mansion
Οικία Χατζηγεωργάκη
Κορνέσιου
Art Gallery
koraï
Mouseiou
Egypt Ave
The Cyprus Museum
Κυπριακό Μουσείο
H&M
Plateia Solomou
Πλατεία Σολωμού
Postal Museum
Ταχυδρομικό Μουσείο
Podocatoro
Bastion
Da Paolo Restaurant
Central Bus
Eleftherias
Square
Solomos Sq
The American Academy
Nicosia Cyprus
Αμερικάνικη Ακαδημία
Kostí Palama
Kawacom's ipanema
Espresso Diagorou
Leventis Art Gallery
Πινακοθήκη Λεβέντη
D'Avilla
Bastion
Προμαχώνας
Νταβίλλα
Μνημείο Ιάκωβου
Πατάτσου
Stasinou
Travel & Express
Pandora
Πανδώρα
Ministry of Labour
& Social Security
Υπουργείο Εργασίας &
Κοινωνικών Ασφαλίσεων
Evagorou
Old GSP
Stadium
Πάρκο (Υπό
σχεδιασμό)
PwC Cyprus - Julia House
Limoncello Deli-Bar
Πρόζακ. Καφενείο
Ministry of Finance
Barrique Wine & Deli
Artisan's Burgerbar
KES College
Pyxida Fish Tavern
Google

Robot Motion Planning



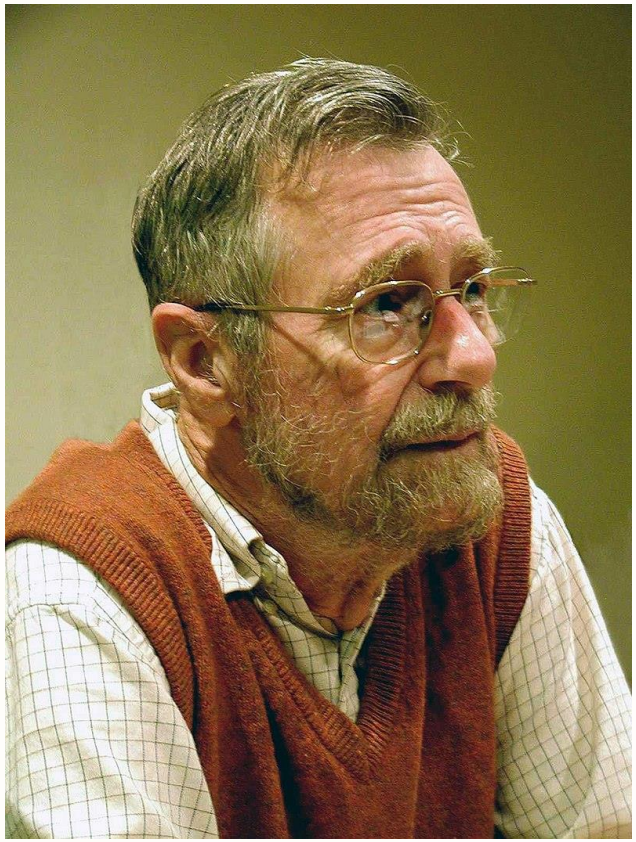
A* in Action - Artificial Intelligence for Robotics: https://youtu.be/qXZt-B7iUyw

Video Games


APM: 487
342
0
15/17
Terran SCV
Terran Marine
Terran SCV
Terran SCV
Terran SCV
Terran SCV
Terran SCV
Terran SCV
Terran SCV
G attack: false G defense: false
[3663:WorkerLoiterAndHarassController16] INFO - Scared of a visible Terran Marine.
Global(D) t: 40 f: 0 n: 0 s:
LOITER_AND_HARASS
MOVE_TO_CANARY
MENU

Edsger W. Dijkstra (1930-2002)

https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

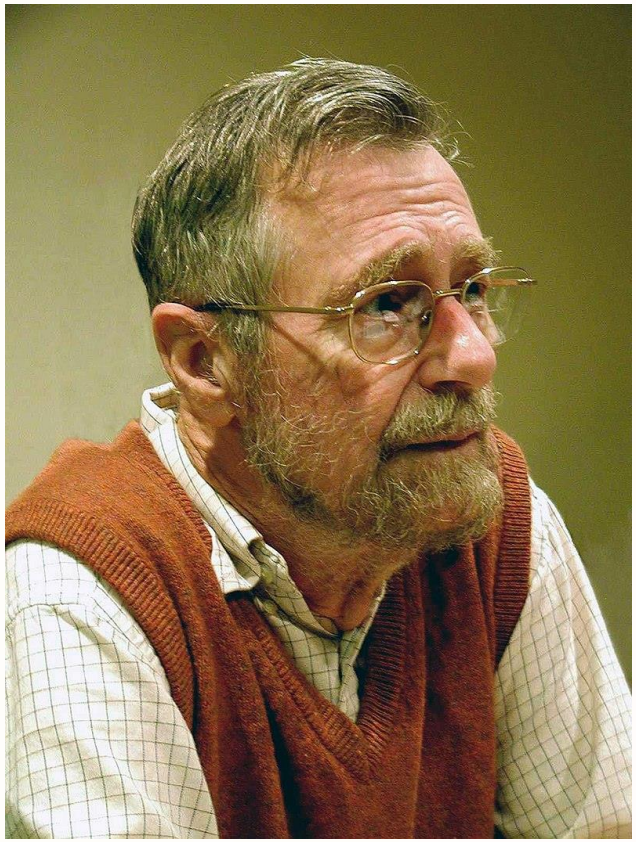


- Dijkstra's algorithm
- DJP algorithm (aka Prim's Algorithm)
- First implementation of ALGOL 60
- Structured programming
- Semaphore
- THE multiprogramming system
- Multithreaded programming
- Concurrent programming
- Principles of distributed computing
- Mutual exclusion
- Call stack
- Fault-tolerant systems

Edsger W. Dijkstra on Dutch TV: https://youtu.be/RCCigccBzIU

Edsger W. Dijkstra (1930-2002)

https://en.wikipedia.org/wiki/Edsger_W._Dijkstra



- Self-stabilizing distributed systems
- Deadly embrace
- Shunting-yard algorithm
- Banker's algorithm
- Dining philosophers problem
- Predicate transformer semantics
- Guarded Command Language
- Weakest precondition calculus
- Smoothsort
- Separation of concerns
- Software architecture

Edsger W. Dijkstra on Dutch TV: https://youtu.be/RCCigccBzIU

Edsger W. Dijkstra Burns

https://en.wikiquote.org/wiki/Edsger_W._Dijkstra

“For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce. “

“FORTRAN, 'the infantile disorder', by now nearly 20 years old, is hopelessly inadequate for whatever computer application you have in mind today: it is now too clumsy, too risky, and too expensive to use.”

“The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offense.” (Back in 1975)

“In the good old days physicists repeated each other's experiments, just to be sure. Today they stick to FORTRAN, so that they can share each other's programs, bugs included.”

“Besides a mathematical inclination, an exceptionally good mastery of one's native tongue is the most vital asset of a competent programmer.”

“It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.”

“A picture may be worth a thousand words, a formula is worth a thousand pictures.”

Edsger W. Dijkstra (Σημερινή Διάλεξη)

Dijkstra's Algorithm: The result of a man minimizing shopping distance **VERY** valuable

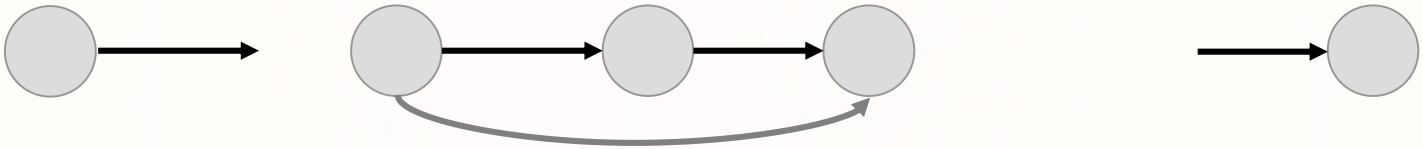
"What is the shortest way to travel from Rotterdam to Groningen, in general: from given city to given city. It is the algorithm for the shortest path, which I designed in about **twenty minutes**. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path. As I said, it was a **twenty-minute invention**. In fact, it was published in '59, three years late. The publication is still readable, it is, in fact, quite nice. One of the reasons that it is so nice was that I designed it without pencil and paper. I learned later that one of the advantages of designing without pencil and paper is that you are almost forced to avoid all avoidable complexities. Eventually that algorithm became, to my great amazement, one of the cornerstones of my fame."

Η δομή της βέλτιστης λύσης

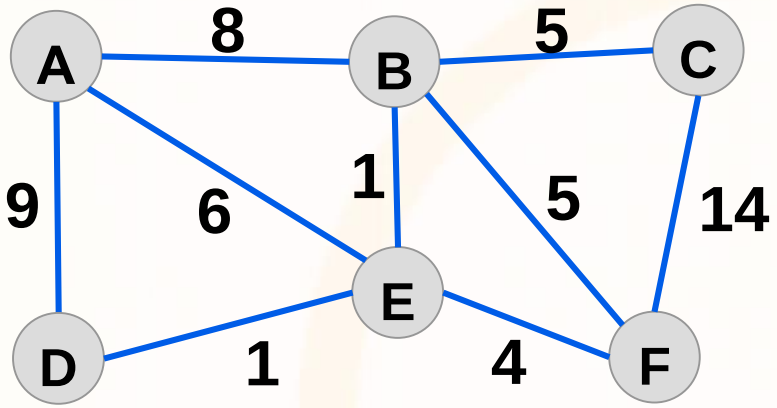
- **Λήμμα 1**
  Έστω κατευθυνόμενος γράφος με βάρη G=(V,E) και έστω

$$p = u \to v_1 \to \ldots \to v_k \to v$$

- το βραχύτερο μονοπάτι μεταξύ των κόμβων u και v.
  Τότε κάθε υπο-μονοπάτι του p είναι βραχύτερο.



- Ας δούμε την λογική πίσω από το Λήμμα.

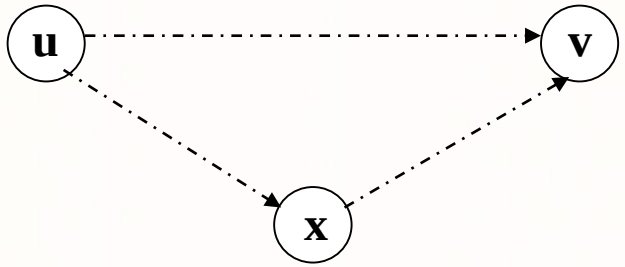


Η δομή της βέλτιστης λύσης (συν.)

Συμβολισμός: Θα γράφουμε **w(u,v)** για το βάρος του βραχύτερου μονοπατιού από την κορυφή u στην κορυφή v.

Λήμμα 2 (Τριγωνική ανισότητα)

Για κάθε τριάδα κορυφών u, v, x, $w(u,v) \leq w(u,x) + w(x,v)$

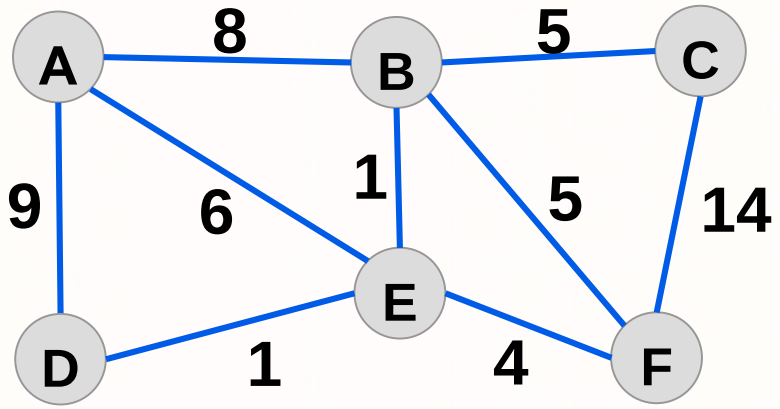


Απόδειξη

- Το ελάχιστο μονοπάτι από τον u στον v δεν είναι μακρύτερο από οποιοδήποτε άλλο μονοπάτι από τον u στο v – εν προκειμένω, το μονοπάτι που παίρνει πρώτα το ελάχιστο μονοπάτι από τον u στον κόμβο x και στη συνέχεια από τον x στον v. □

Ανασκόπηση του Αλγόριθμου Dijkstra

- Έστω ότι θέλουμε να βρούμε το/α **βραχύτερα μονοπάτια** από κάποιο κόμβο **A** προς όλους τους υπόλοιπους κόμβους σε κάποιο γράφο **G(V,E)**

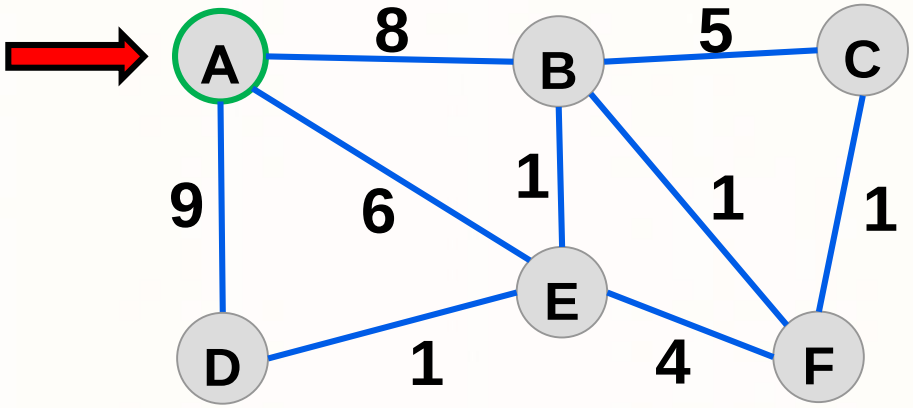


- Ο αλγόριθμος του Dijkstra είναι ένας **single source shortest path algorithm** (δηλαδή αναφέρεται συγκεκριμένα σε μια κορυφή εκκίνησης) για γράφους με **μη-αρνητικά βάρη**. Βρίσκει:

1. Ποιο είναι το κόστος του βραχύτερου μονοπατιού από **D** προς C
2. Ποιο είναι το ακριβές μονοπάτι

Παράδειγμα Εκτέλεσης Αλγορίθμου Dijkstra 1/4

- Πάρε σαν παράμετρο κάποια κορυφή, έστω η κορυφή Α



S: Το σύνολο όλων των κόμβων που έχουμε βρει το ελάχιστο μονοπάτι

- Αρχικοποίησε ένα **πίνακα distance** και θέσε την απόσταση όλων των κορυφών από την κορυφή Α είναι ίση με ∞.

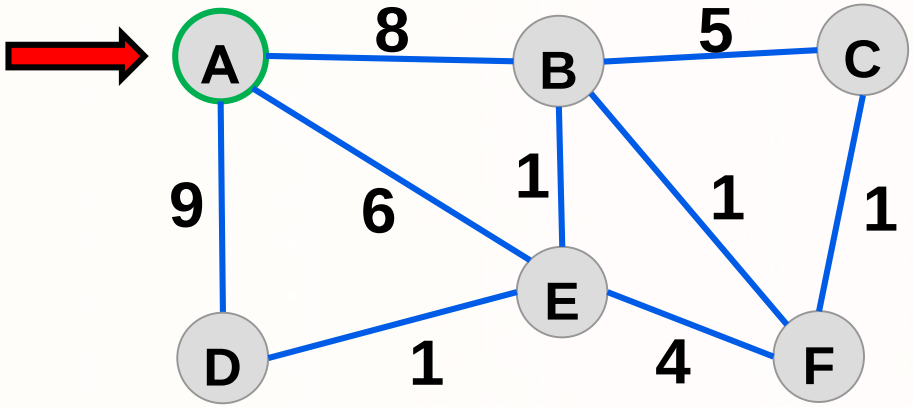
Εκτέλεση	S	d(A)	d(B)	d(C)	d(D)	d(E)	d(F)
1	S=∅	0	∞	∞	∞	∞	∞

- Αρχικοποίησε ένα **πίνακα visited** όπου θα σημειώνει **από ποιες κορυφές έχουμε περάσει ήδη**

Εκτέλεση	S	v(A)	v(B)	v(C)	v(D)	v(E)	v(F)
1	S=∅	1	0	0	0	0	0

Παράδειγμα Εκτέλεσης Αλγορίθμου Dijkstra 2/4

- Αρχικοποίησε τον πίνακα distance με τις ακμές της κορυφής εισόδου.

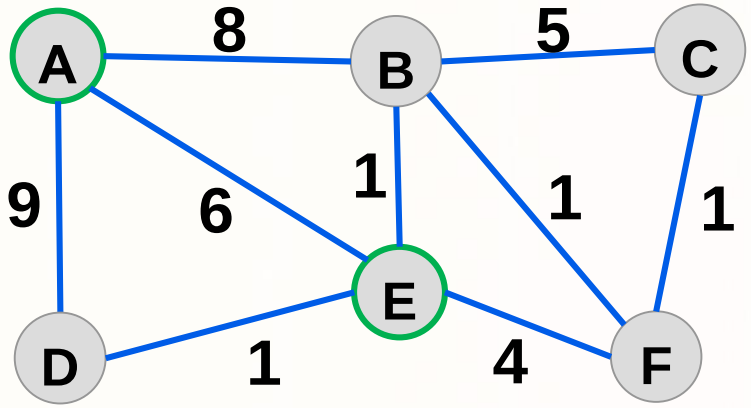


Κορυφές που έχουμε επισκεφθεί (visited=1)

Εκτέλεση	S	d(A)	d(B)	d(C)	d(D)	d(E)	d(F)
1	S=∅	0	∞	∞	∞	∞	∞
2	S={A}	0	8	∞	9	6 ⇐	∞

- Σε κάθε βήμα του αλγόριθμου, **επίλεξε** «άπληστα» την **κορυφή με την ελάχιστη απόσταση που δεν έχεις επισκεφθεί**

Παράδειγμα Εκτέλεσης Αλγορίθμου Dijkstra 3/4



Κορυφές που έχουμε επισκεφθεί (visited=1)

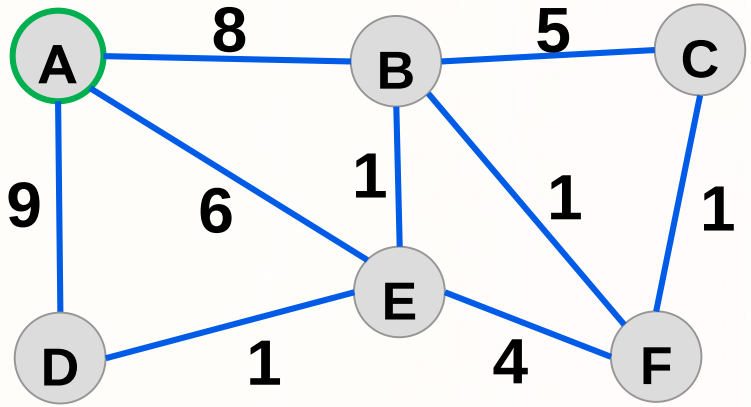
Επιλογή της ελάχιστης απόστασης

X Μείωση από την προηγούμενη απόσταση

Εκτέλεση	S	d(A)	d(B)	d(C)	d(D)	d(E)	d(F)
1	S=∅	0	∞	∞	∞	∞	∞
2	S={**A**}	0	8	∞	9	6	∞
3	S={A,**E**}	0	7	∞	7	6	10

- Η E (απόσταση=6) έχει ακμές με **B,D**(6+1=7) και **F**(6+4=10)
- **Αφού οι αποστάσεις αυτές είναι μικρότερες από τις υπάρχουσες τότε ενημερώνεται ο πίνακας ανάλογα.**

Παράδειγμα Εκτέλεσης Αλγορίθμου Dijkstra 4/4



Κορυφές που έχουμε επισκεφθεί (visited=1)

Επιλογή της ελάχιστης απόστασης

X Μείωση από την προηγούμενη απόσταση

Εκτέλεση	S	d(A)	d(B)	d(C)	d(D)	d(E)	d(F)
1	S=∅	0	∞	∞	∞	∞	∞
2	S={**A**}	0	8	∞	9	6	∞
3	S={A,**E**}	0	7	∞	7	6	10
4	S={A,E,**B**}	0	7	12	7	6	8
5	S={A,E,B,D}	0	7	12	7	6	8
6	S={A,E,B,D,F}	0	7	9	7	6	8
7	S={A,E,B,D,F, C}	0	7	9	7	6	8

Υλοποίηση του Αλγόριθμου Dijkstra

```
// A: Σημείο Εκκίνησης. G(V,E): Ο Γράφος
dijkstra( G(V,E), vertex A){

   distance[|V|]={∞}; // Απόσταση κορυφής i από κορυφή A
   visited[|V|]={};   // Σημειώνει αν περάσαμε η όχι από κορυφή
   count = 0;         // Μετρητής κορυφών που επισκεφθήκαμε

  // Αρχικοποίηση σημείου εκκίνησης
  distance[A]=0; visited[A]=1;

   while (count < |V|){     // σε χρόνο O(|V|)
        // Προχωρούμε στον επόμενο κόμβο
        u=minVertex(V);     // Άπληστη επιλογή σε χρόνο O(|V|)

        // Ενημέρωση αποστάσεων γειτόνων προς A
        για κάθε γείτονα v του u {
            if (distance[v] > distance[u] + w(u,v))
               distance[v] = distance[u] + w(u,v);
        }
        count++;
   }
}
```

Συνολικός Χρόνος Εκτέλεσης: $O(|V|^2)$

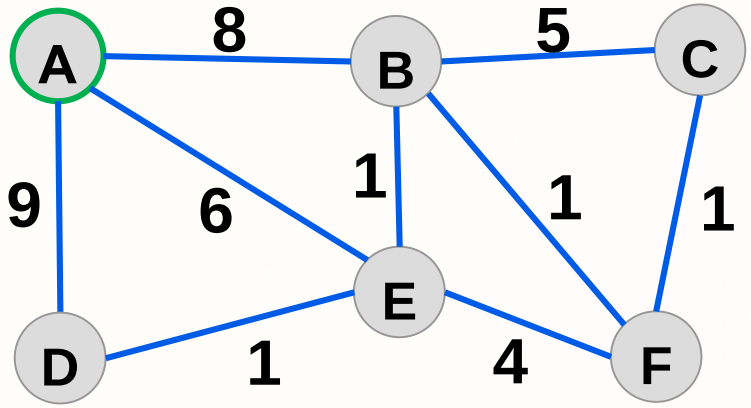
Dijkstra για εύρεση του Βραχύτερου Μονοπατιού

- Αν εκτός από το μήκος του μονοπατιού μας ενδιαφέρει και το ακριβές μονοπάτι (οι κόμβοι του) τότε μπορούμε να ακολουθήσουμε την ίδια βασική ιδέα με την εξής προσθήκη

- Για κάθε κορυφή αποθηκεύουμε σε ένα πίνακα P, τον γείτονα που θα μας δώσει το βραχύτερο μονοπάτι προς τον κόμβο εκκίνησης.

- Σε αυτή την περίπτωση μπορούμε με τον τερματισμό του αλγόριθμου να κατασκευάσουμε από τον πίνακα P το μέγιστο μονοπάτι από τον κόμβο εκκίνησης προς κάποιον άλλο κόμβο X

Αλγόριθμος Dijkstra για εύρεση Βραχ. Μονοπατιού



Κορυφές που έχουμε επισκεφθεί (visited=1)

Επιλογή της ελάχιστης απόστασης

X Μείωση από την προηγούμενη απόσταση

Εκτέλεση	S	d(A)	d(B)	d(C)	d(D)	d(E)	d(F)
1	S=∅	0,-	∞,-	∞,-	∞,-	∞,-	∞,-
2	S={**A**}	0,-	8,A	∞	9,A	6,A	∞
3	S={A,**E**}	0,-	**7,E**	∞	**7,E**	6,A	10,E
4	S={A,E,**B**}	0,-	7,E	12,B	7,E	6,A	**8,B**
5	S={A,E,B,D}	0,-	7,E	12,B	7,E	6,A	8,B
6	S={A,E,B,D,F}	0,-	7,E	**9,F**	7,E	6,A	8,B
7	S={A,E,B,D,F, C}	0,-	7,E	9,F	7,E	6,A	8,B

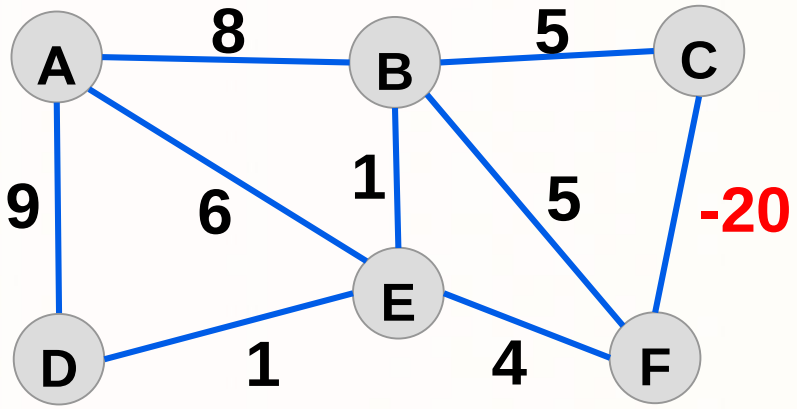
Υλοποίηση του Αλγόριθμου Dijkstra

```
// A: Σημείο Εκκίνησης. G(V,E): Ο Γράφος
dijkstra( G(V,E), vertex A){
   distance[|V|]={∞}; // Απόσταση κορυφής i από κορυφή A
   visited[|V|]={};   // Σημειώνει αν περάσαμε η όχι από κορυφή
   shortest[|V|]={};   // Σημειώνει το βραχύτερο μονοπάτι
   count = 0;          // Μετρητής κορυφών που επισκεφθήκαμε
  // Αρχικοποίηση σημείου εκκίνησης
  distance[A]=0; visited[A]=1;
   while (count < |V|){     // σε χρόνο O(|V|)
        // Προχωρούμε στον επόμενο κόμβο
        u=minVertex(V);     // Άπληστη επιλογή σε χρόνο O(|V|)
        // Ενημέρωση αποστάσεων γειτόνων προς A
        για κάθε γείτονα v του u {
            if (distance[v] > distance[u] + w(u,v)) {
               distance[v] = distance[u] + w(u,v);
               shortest[v] = u; }
        }
        count++;
   }
}
```

Αρνητικά Βάρη στον Αλγόριθμο του Dijkstra

- Σημειώστε ότι όταν υπάρχουν αρνητικά βάρη, τότε είναι δυνατό αυτά να μας θέσουν την μικρότερη απόσταση, **αναδρομικά, ίση με -∞.**

Παράδειγμα:



Ποια είναι η βραχύτερη απόσταση μεταξύ (B, C);

- Πρώτη Εκτέλεση 5: B=>C
- Δεύτερη Εκτέλεση -5: B=>C=>F=>B=>C
- Τρίτη Εκτέλεση -15: B=>C=>F=>B=>C=>F=>B=>C
- .....

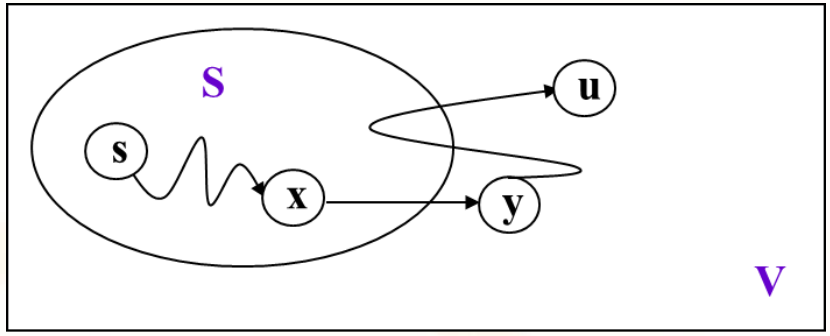
Απόδειξη ορθότητας

Θεώρημα 1 Όταν μια κορυφή u 'μπαίνει' στο S, τότε d[u] = w(s,u).

Απόδειξη

- Υποθέτουμε, για να φτάσουμε σε αντίφαση, ότι η u είναι η πρώτη κορυφή η οποία, κατά την εκτέλεση του αλγόριθμου, μπαίνοντας στο S έχει d[u] μεγαλύτερο (προσέξτε Λήμμα 2) από το βάρος του βραχύτερου μονοπατιού μεταξύ της u και της s, δηλ., **d[u] > w(s,u)**.

- Έστω y η `πρώτη΄ κορυφή στο V-S που ανήκει στο βραχύτερο μονοπάτι από την s στη u.



- Αφού η x μπήκε στο S πριν από την u, d[x]=w(s,x).

- Επίσης, με την εισαγωγή του x στο S, το d[y]=d[x]+w(x,y), το οποίο είναι το κόστος του υπομονοπατιού στο σχήμα.

Απόδειξη ορθότητας

- Αφού το μονοπάτι από το s στο u, είναι βραχύτερο, τότε από τη δομή βέλτιστης λύσης συνεπάγεται ότι το υπομονοπάτι $s \Rightarrow x \rightarrow y$ από το s στο y είναι επίσης βραχύτερο. Άρα $d[y] = w(s,y)$.
- Έτσι:

$d[u] > w(s,u)$ (αρχική υπόθεση)

$= w(s,y) + w(y,u)$ (δομή βέλτιστης λύσης)

$= d[y] + w(y,u)$ ($d[y] = w(s,y)$)

$\geq d[y]$ (τα κόστα είναι ≥ 0)

- Αφού $d[u] > d[y]$, ο αλγόριθμος θα διάλεγε και θα εισήγαγε τη y στο S και όχι τη u. **Αντίφαση!** □

<u>Λήμμα 2</u> Καθ' όλη τη διάρκεια του αλγόριθμου $d[u] \geq w(s,u)$.
Η απόδειξη είναι παρόμοια με αυτή του Λήμματος 1.

- Χρόνος Εκτέλεσης: $(|V| + |E|) \cdot \log|V|$

Dijkstra

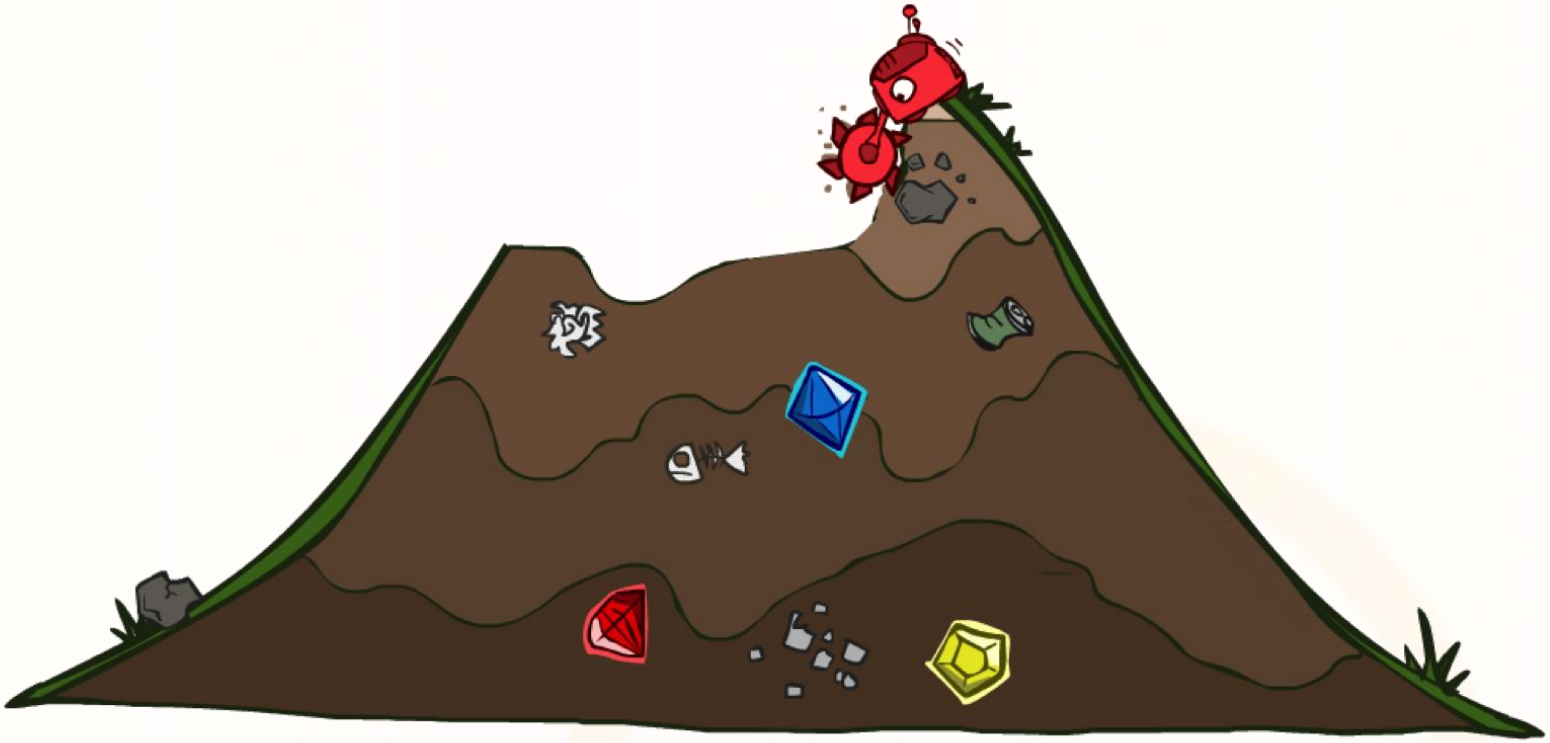


Image from: https://inst.eecs.berkeley.edu/~cs188/

Greedy

A* Search: Informed Search

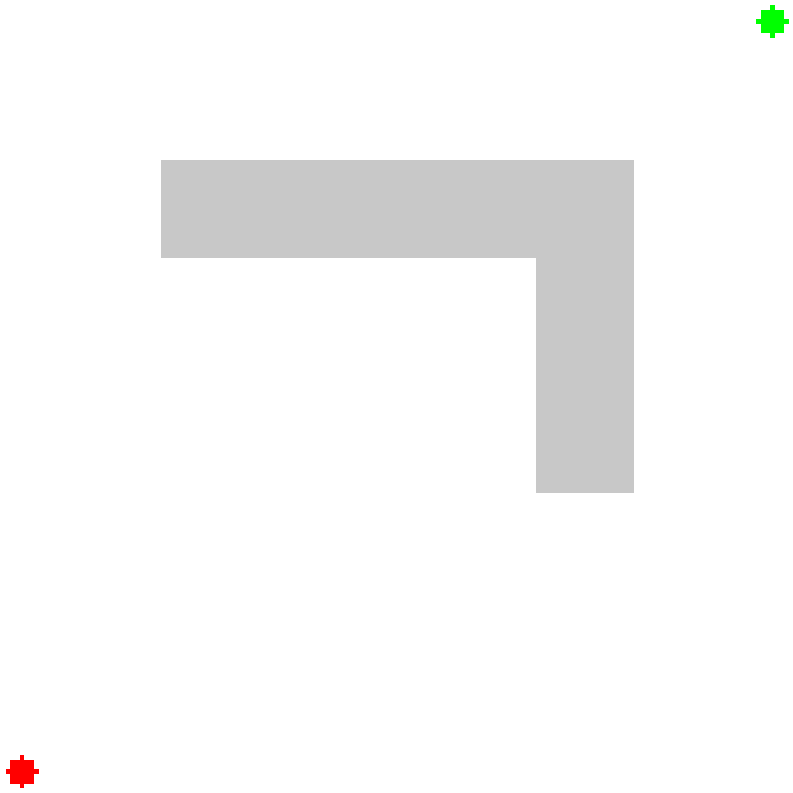
In A* we add a **heuristic** on how close we move towards a goal vertex to avoid examining unnecessary vertices (i.e., consider asking Google Maps the closest route to Limassol from Nicosia. Dijkstra would consider all places in a very large radius whereas our target is South-West!

Image from: https://inst.eecs.berkeley.edu/~cs188/

Dijkstra vs A*

Animations from:
https://en.wikipedia.org/wiki/Dijkstra's_algorithm
https://en.wikipedia.org/wiki/A*_search_algorithm

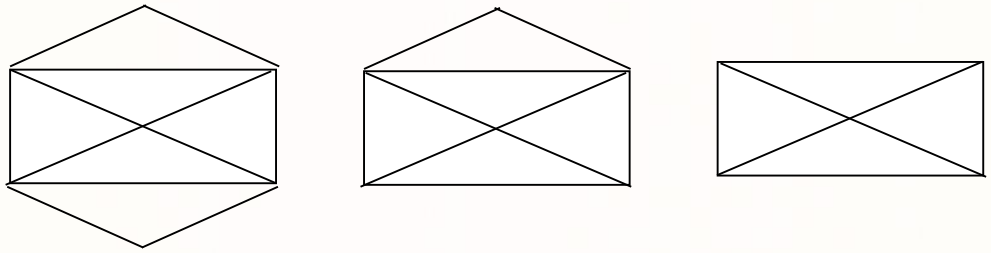
Dijkstra



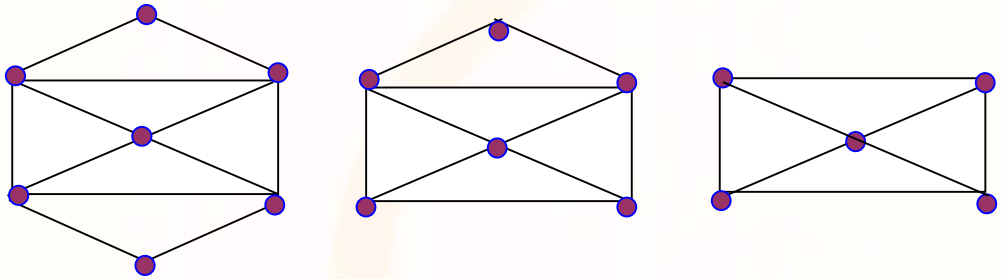
A*

Μονοπάτια Euler

- (Παράδειγμα Εφαρμογής κατά βάθους διερεύνησης)
- Σπαζοκεφαλιά: μπορούμε να δημιουργήσουμε τα πιο κάτω σύνολα γραμμών ζωγραφίζοντας κάθε γραμμή ακριβώς μια φορά χωρίς να σηκώσουμε την πένα από το χαρτί;



- Το πρόβλημα μπορεί να μετατραπεί σε πρόβλημα γράφων: Κτίζουμε γράφο G του οποίου κόμβοι είναι τα σημεία όπου διασταυρώνονται γραμμές του σχεδίου και ακμή μεταξύ δύο κόμβων υπάρχει αν υπάρχει γραμμή μεταξύ των αντίστοιχων σημείων του σχεδίου.

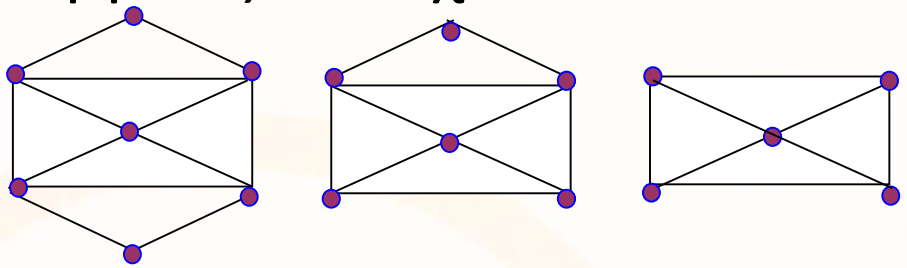


Μονοπάτια Euler

- Το πρόβλημα τώρα είναι να διακριβώσουμε την ύπαρξη μονοπατιού που περνά από κάθε ακμή ακριβώς μια φορά. Τέτοια μονοπάτια ονομάζονται ***μονοπάτια Euler***, από το μαθηματικό Euler ο οποίος έλυσε το πρόβλημα το 1732.
- Ένας γράφος με n κόμβους έχει μονοπάτι Euler αν και μόνο αν
  1. είναι συνεκτικός,
  2. τουλάχιστον n-2 από τους κόμβους του έχουν

     Degree mod 2 = 0

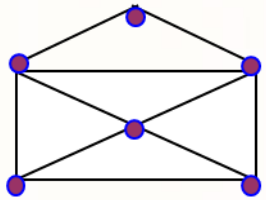
     όπου Degree ενός κόμβου v είναι ο αριθμός των ακμών που ξεκινούν απ' αυτόν.



- Η δεύτερη ιδιότητα οφείλεται στο ότι για όλους εκτός από τον πρώτο και τον τελευταίο κόμβο κάθε φορά που το μονοπάτι 'μπαίνει' σε ένα κόμβο θα πρέπει και να 'βγει'.
- Πόσα μονοπάτια Euler έχει ένας γράφος;

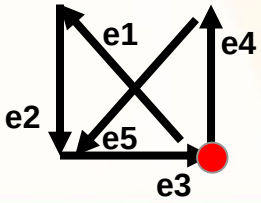
Εύρεση Μονοπατιού Euler

- Πως μπορούμε να βρούμε μονοπάτια Euler; Υπάρχει αλγόριθμος βασισμένος σε DFS ο οποίος πετυχαίνει το στόχο σε χρόνο γραμμικό.
- Έστω ότι γνωρίζουμε την ύπαρξη μονοπατιού Euler σε ένα γράφο. Εφαρμόζουμε τα εξής βήματα:
  1. διαλέγουμε μια από τις κορυφές για τις οποίες Degree mod 2 = 1 αν τέτοια κορυφή υπάρχει, διαφορετικά οποιαδήποτε άλλη κορυφή.
     Ποιες είναι υποψήφιες κορυφές εκκίνησης στον γράφο στα δεξιά;



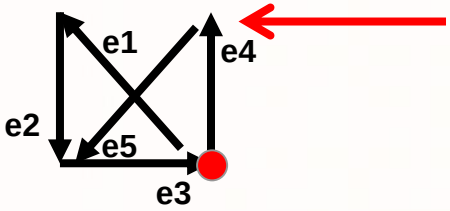
  2. Εφαρμόζουμε κατά βάθος διερεύνηση μέχρις ότου να μην μπορούμε να προχωρήσουμε. Έστω ότι παίρνουμε το μονοπάτι:

$$s = e_1 \ldots e_n$$



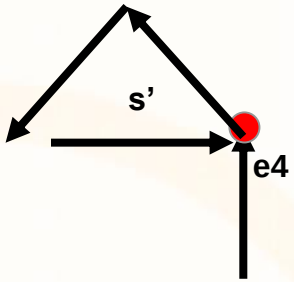
Εύρεση Μονοπατιού Euler (συν.)

3. Βρίσκουμε την πρώτη ακμή e =(u,v) στο μονοπάτι s στην οποία ο αρχικός κόμβος, u, έχει παιδιά τα οποία δεν έχουν ήδη εξερευνηθεί και από εκεί ξεκινούμε DFS διερεύνηση στο μέρος του γράφου που παραμένει ανεξερεύνητο.



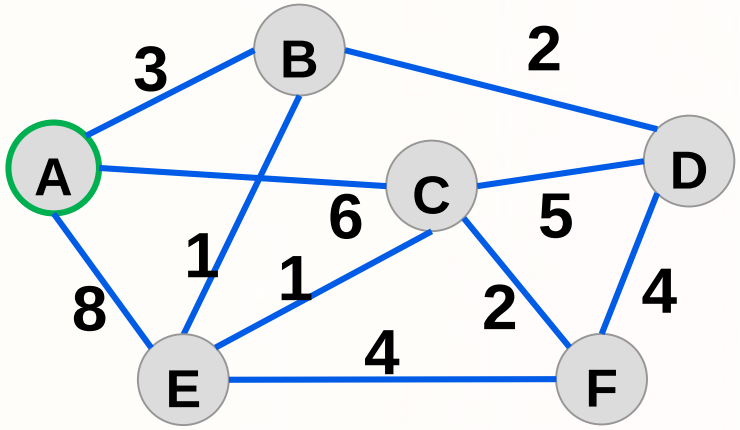
4. Έστω ότι λαμβάνουμε το μονοπάτι s΄. Προσθέτουμε το s΄ στο s ως εξής:

$$s = e_1 \ldots e_{i-1} s' e_i \ldots e_n$$



5. Επαναλαμβάνουμε το βήμα 3 μέχρις ότου να εξερευνηθεί ολόκληρος ο γράφος και επιστρέφουμε το μονοπάτι s.

Άσκηση: Εκτελέστε τον αλγόριθμο του Dijkstra



Κορυφές που έχουμε επισκεφθεί (visited=1)

Επιλογή της ελάχιστης απόστασης

X

Μείωση από την προηγούμενη απόσταση

Εκτέλεση	S	d(A)	d(B)	d(C)	d(D)	d(E)	d(F)
1	∅	0,-	∞,-	∞,-	∞,-	∞,-	∞,-