Επίλυση Προβλημάτων με Greedy Αλγόριθμους

Περίληψη

- Επίλυση προβλημάτων χρησιμοποιώντας Greedy Αλγόριθμους
- Ελάχιστα Δέντρα Επικάλυψης
  - Αλγόριθμος του Prim
  - Αλγόριθμος του Kruskal
- Πρόβλημα Ελάχιστης Απόστασης
  - Dijkstra’s algorithm
- Συμπίεση Αρχείων
  - Huffman Codes
- Επίλυσης μη γραμμικών εξισώσεων

Greedy Αλγόριθμοι

- Χρησιμοποιούνται περισσότερο για βελτιστοποίηση
- Σε κάθε βήμα, οι αλγόριθμοι αυτού του τύπου
  - Πηγαίνουν σε μια εφικτή λύση (feasible solution)
  - Η λύση αυτή είναι «τοπικά» βέλτιστη (locally optimal).
    - Δεδομένης της λύσης x_k στην οποία βρίσκεται ο αλγόριθμος μετά την επανάληψη k, στην επόμενη επανάληψη θα πάει σε μια λύση στην γειτονιά της x_k η οποία είναι καλύτερη απ' ότι η λύση στη x_k.
  - Δεν επιστρέφουν σε προηγούμενη λύση (δεν έχουν μνήμη).
- Χαρακτηριστικά και Ιδιότητες των αλγορίθμων αυτών
  - Οι αλγόριθμοι αυτού του τύπου, σε κάθε βήμα τους πηγαίνουν σε μια λύση η οποία είναι καλύτερη από την προηγούμενη.
  - Το μεγαλύτερο πρόβλημα αυτών των αλγορίθμων είναι ότι μπορούν εύκολα να εγκλωβιστούν σε τοπικά βέλτιστές λύσεις.

Πρόβλημα υπολογισμού νομισμάτων

- Υποθέστε πως υπάρχουν κέρματα 1c, 5c, 10c.
- Περιγράψτε ένα αλγόριθμο για τον υπολογισμό του συνδυασμού των νομισμάτων που πρέπει κάποιος να επιστρέψει για ρέστα έτσι που να ελαχιστοποιείται ο αριθμός των κερμάτων.
- Παράδειγμα:
  - Θέλετε να επιστρέψετε 18c. Πως θα το καταφέρεται;
  -
  -
  -

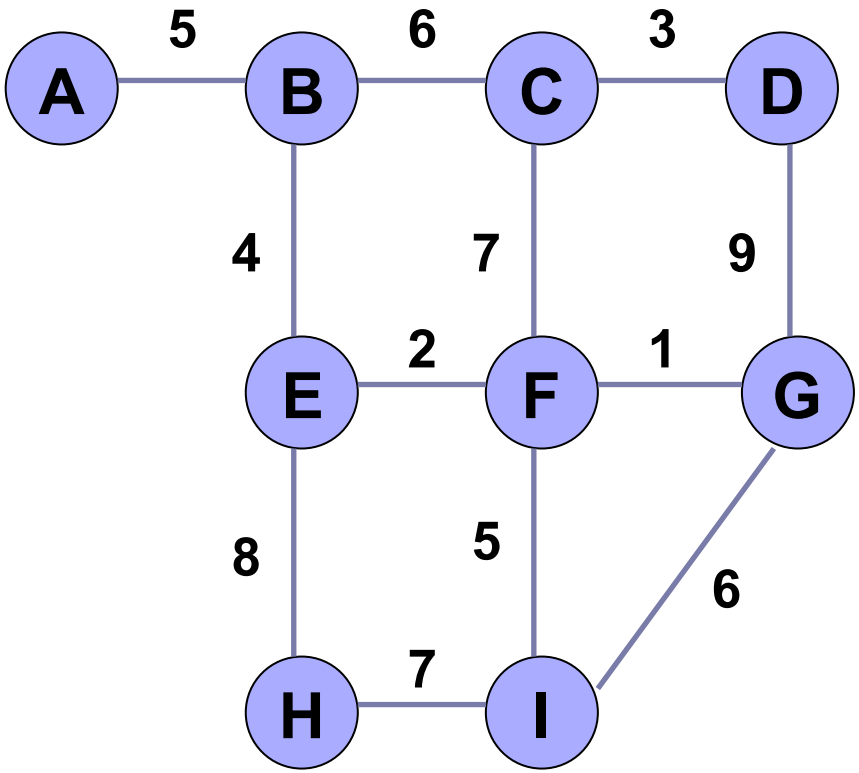
Αλγόριθμος

```
change(x)
  N=[N10c, N5c, N1c];//coins 10c, 5c, 1c.
  N10c= floor(x/10);
  x= x – 10*N10c;
  N5c= floor(x/5);
  N1c= x – 5*N5c;
  return N;
```

- Ο αλγόριθμος δουλεύει για οποιοδήποτε συνδυασμό κερμάτων;
-
  -
  -

Πρόβλημα εγκατάστασης δικτύου ελάχιστου κόστους

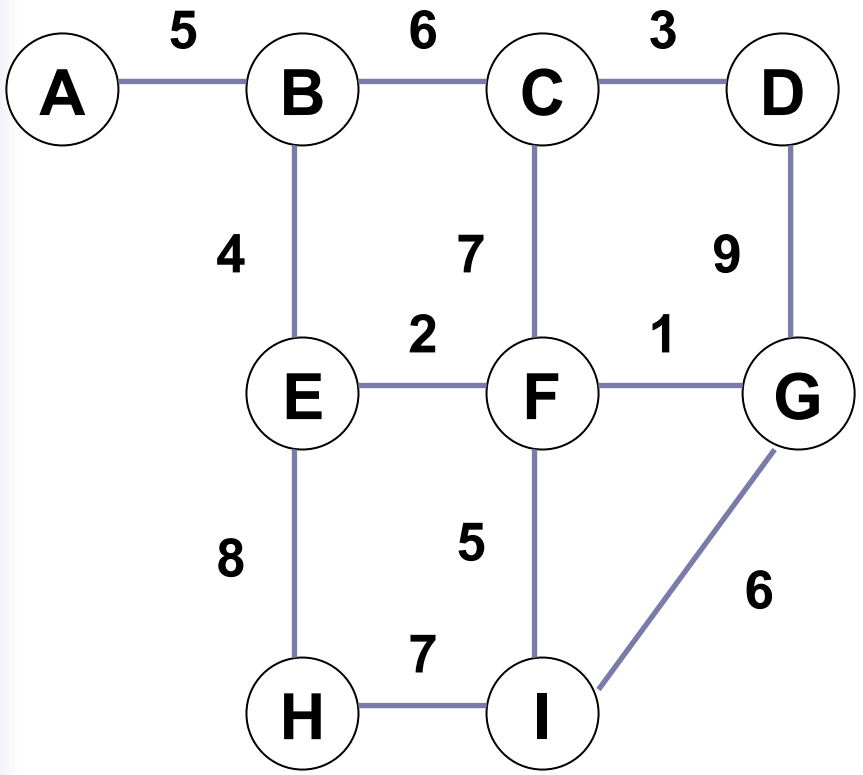
- Υποθέστε ότι θέλετε να συνδέσετε 9 σημεία σε ένα δίκτυο έτσι που να μπορούν όλα να επικοινωνήσουν μεταξύ τους
- Ένα κόμβος μπορεί να επικοινωνήσει με κάποιον άλλο είτε απευθείας είτε μέσω κάποιου άλλου κόμβου εφόσον υπάρχει καλώδιο που τους συνδέει.
- Οι αποστάσεις μεταξύ των κόμβων φαίνονται στον γράφο
- Το κόστος καλωδίωσης είναι 100 ευρώ το μέτρο.
- Πως θα συνδέσετε τους κόμβους για να ελαχιστοποιήσετε το κόστος καλωδίωσης;



Δέντρα Επικάλυψης (Spanning Trees)

- **Δέντρο Επικάλυψης**
  - Είναι ένας γράφος που περιέχει όλες τις κορυφές ενός γράφου αλλά δεν έχει κύκλους (είναι δηλαδή δέντρο).
- **Ελάχιστο Δέντρο Επικάλυψης**
  - Είναι το δέντρο επικάλυψης του οποίου το άθροισμα των βαρών (weights) των ακμών που συμπεριλαμβάνονται στο δέντρο επικάλυψης είναι ελάχιστο.

Αλγόριθμος του Prim

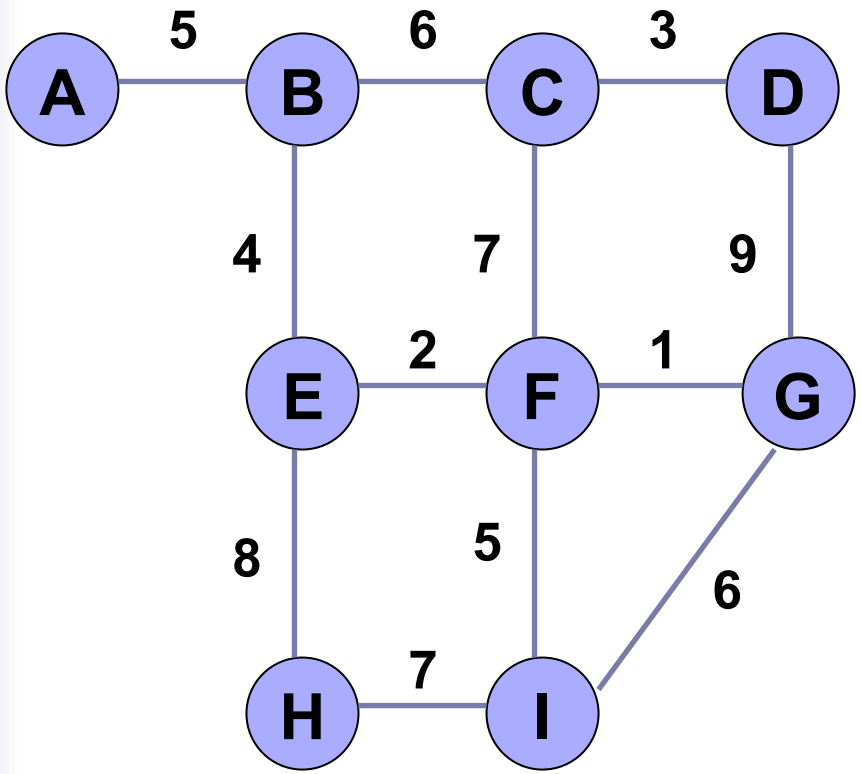


- Οι κόμβοι χωρίζονται σε καταχωρημένους και μη.
- Αρχικά κανένας κόμβος δεν είναι «καταχωρημένος» και επιλέγεται τυχαία ο πρώτος κόμβος που θα καταχωρηθεί.
- Σε κάθε βήμα προσθέτουμε στο σύνολο των καταχωρημένων κόμβων τον μη-καταχωρημένο κόμβο που μπορεί να συνδεθεί με το μικρότερο κόστος.

Αλγόριθμος του Prim

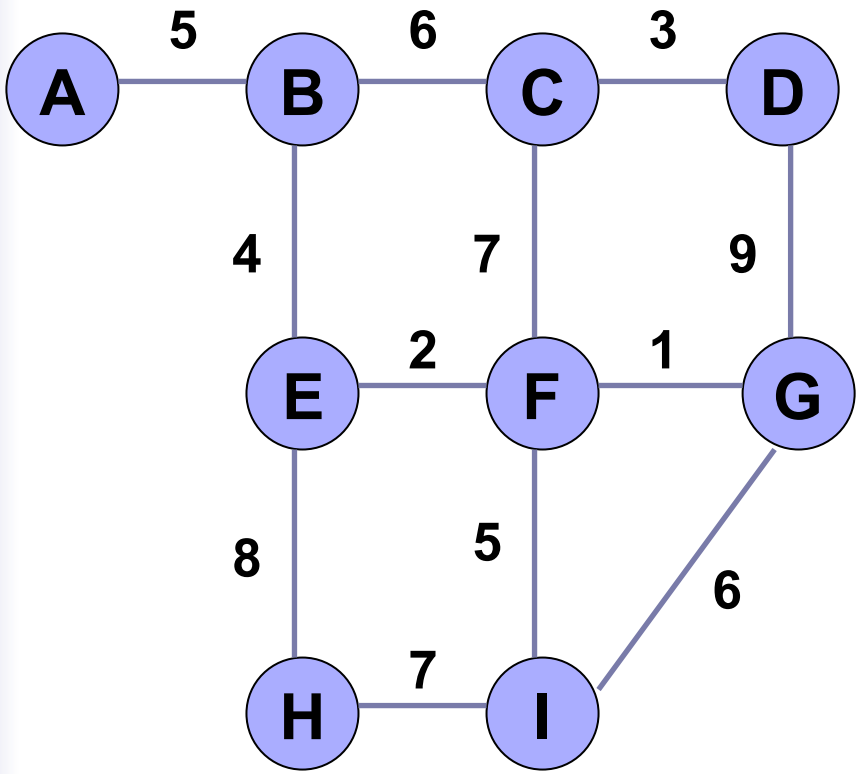
```
Prim(G=(V,E))
  VT= {v0}, ET= empty;
  for i=1 to |V|-1
     Find the minimum weight edge e*=(u*,v*)
     such that u* in VT and v* in V-VT
     VT= VT U {v*};
     ET= ET U {e*};
  return ET;
```

Ορθότητα Αλγόριθμου του Prim



- Απόδειξη χρησιμοποιώντας επαγωγή.
- Στο αρχικό βήμα ο αλγόριθμος είναι ορθός αφού πρέπει να περιλαμβάνει οποιοδήποτε κόμβο.
- Υποθέτουμε πως το δέντρο που δημιουργήθηκε μέχρι το βήμα k-1 είναι μέρος ενός βέλτιστου δέντρου και ελέγχουμε το δέντρο μεγέθους k που δημιουργήθηκε βάση του αλγόριθμου.
- Η υπόθεση αυτή ισχύει, γιατί;

Αλγόριθμος του Kruskal



- Όλες οι ακμές ταξινομούνται σύμφωνα με το βάρος τους.
- Επιλέγονται οι πρώτες |V|-1 ακμές οι οποίες δεν κλείνουν κάποιο κύκλο.

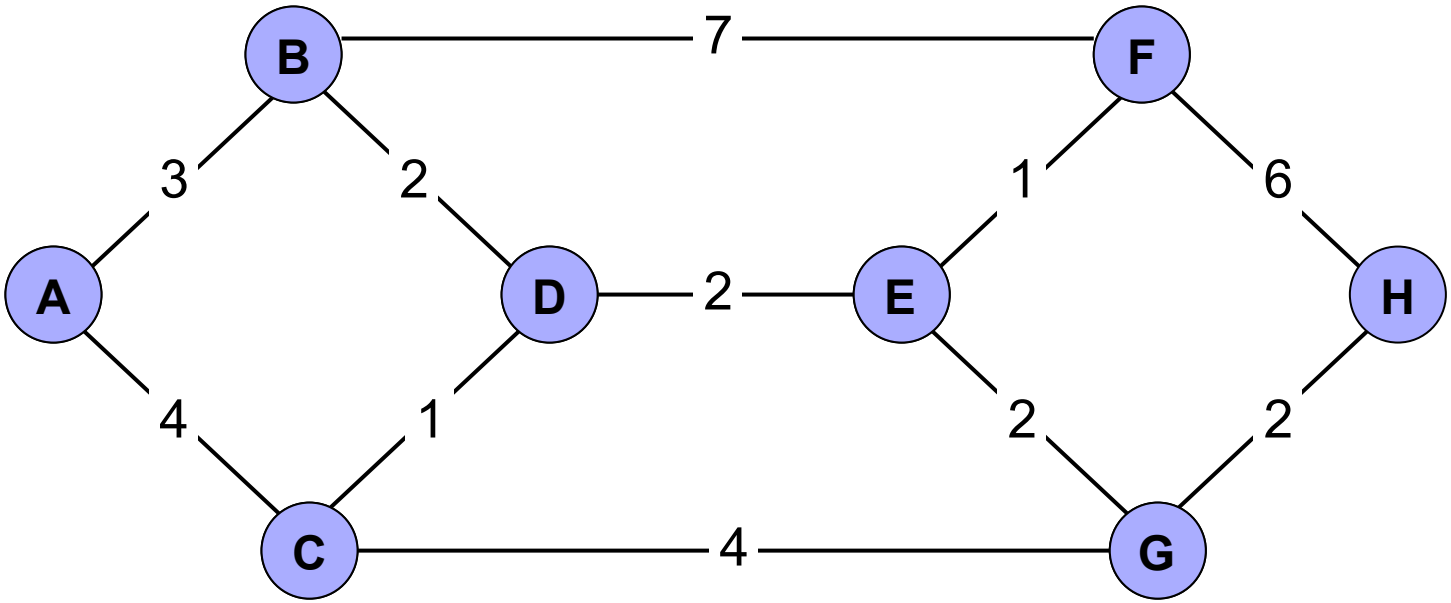
Edge	w	Edge	w
FG	1	BC	6
EF	2	GI	6
CD	3	CF	7
BE	4	HI	7
AB	5	EH	8
FI	5	DG	9

Αλγόριθμος του Kruskal

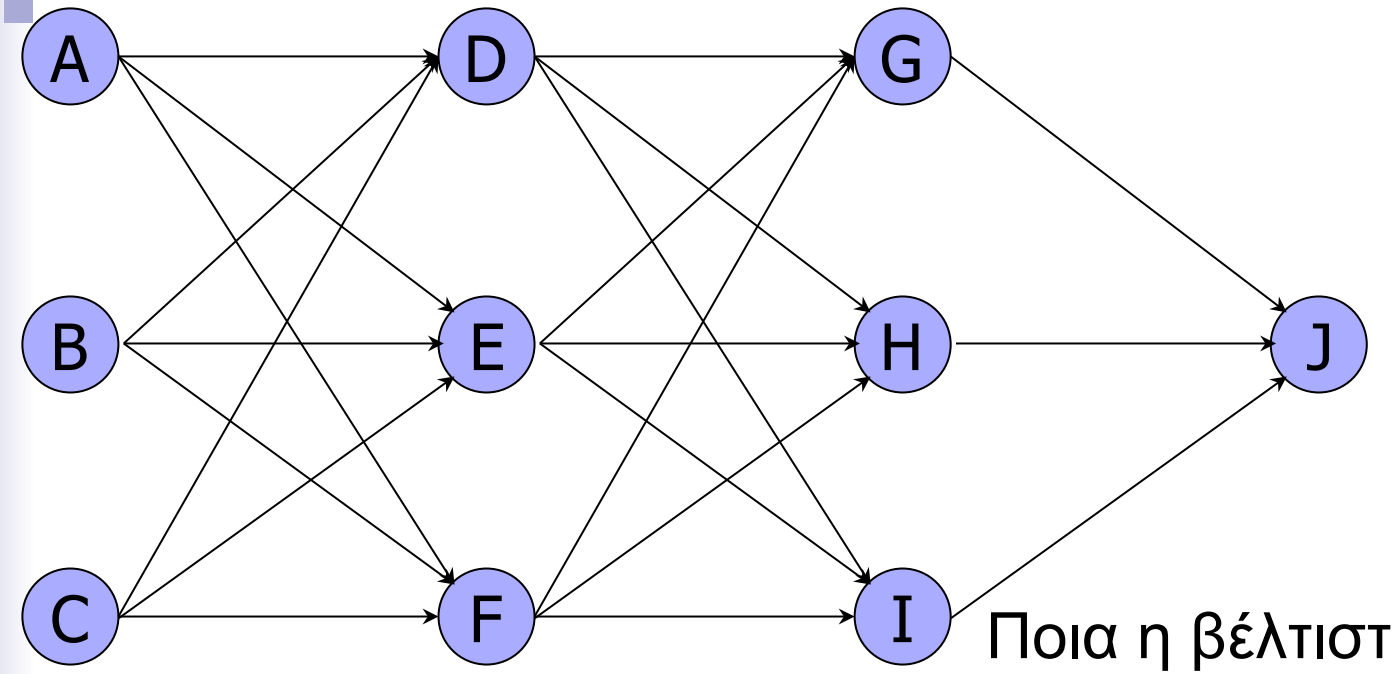
```
Kruskal(G=(V,E))
   sort(E);
   ET= empty;
   count= 0;
   while count < |V|-1
      e= next edge in the list;
      if ET U {e} has no cycle
            ET= ET U {e};
            count= count+1;
   return ET;
```

Πρόβλημα Ελάχιστης Απόστασης

- Δεδομένου ενός γράφου (G,V) με βάρη στις ακμές και μίας πηγής s υπολογίστε:
  - Το ελάχιστο απλό μονοπάτι από τον s σε κάποιο άλλο κόμβο u.
  - Το ελάχιστο απλό μονοπάτι από τον s σε όλους τους υπόλοιπους κόμβους.
    - Απλό μονοπάτι: μονοπάτι χωρίς κύκλους.



Ιδιότητα Ελάχιστου Μονοπατιού



Ποια η βέλτιστη διαδρομή E→J?

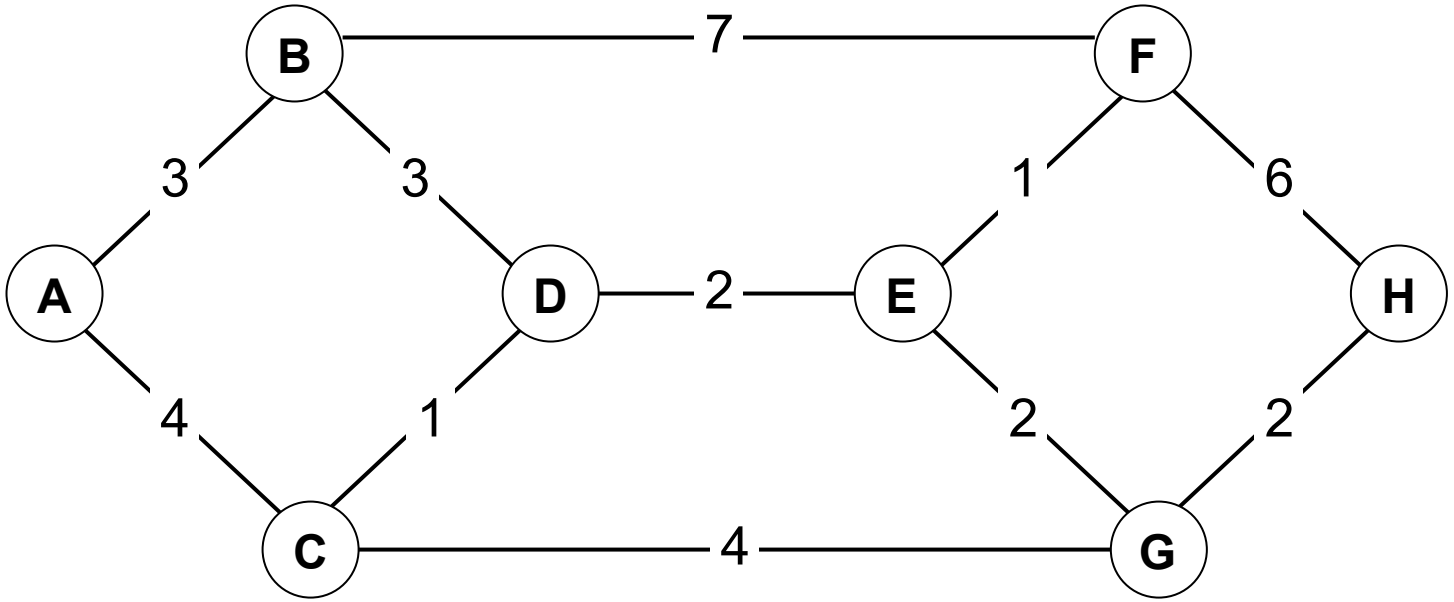
Υποθέστε πως η ελάχιστη διαδρομή από κάποιο κόμβο v_0 στον κόμβο v_n περνά από το μονοπάτι v_i, v_{i+1}, ...,v_j, τότε το μονοπάτι v_i, v_{i+1}, ...,v_j, αποτελεί το ελάχιστο μονοπάτι από τον v_i στον v_j

Πρόβλημα Ελάχιστης Απόστασης (Dijkstra's algorithm)

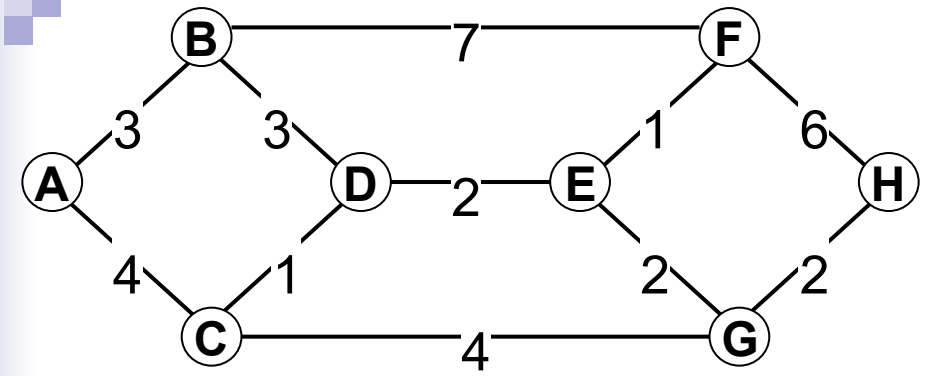
- Οι κόμβοι χωρίζονται σε καταχωρημένους και μη
  - Αρχικά όλοι οι κόμβοι είναι μη καταχωρημένοι
- Ορίζουμε τον παραλήπτη (ή αποστολέα) σαν **ενεργό κόμβο** (working node)
  - Τον σημαδεύουμε με το κόστος μέχρι τον παραλήπτη (ή αποστολέα)
  - Υπολογίζουμε το κόστος για όλους τους κόμβους που είναι συνδεδεμένοι με τον ενεργό κόμβο.
- Επιλέγουμε (από τους μη καταχωρημένους κόμβους) τον κόμβο με το μικρότερο κόστος, τον καταχωρούμε και τον ονομάζουμε σαν τον νέο ενεργό κόμβο.
  - Υπολογίζουμε το κόστος για όλους τους κόμβους που είναι συνδεδεμένοι με τον ενεργό κόμβο
  - Επαναλαμβάνουμε το τελευταίο βήμα μέχρι να καταχωρήσουμε όλους τους κόμβους.

Πρόβλημα Ελάχιστης Απόστασης (Dijkstra's algorithm)

Βρέστε το μονοπάτι με το μικρότερο κόστος από τον A στον H



Πρόβλημα Ελάχιστης Απόστασης (Dijkstra's algorithm)



	B	C	D	E	F	G	H

Αλγόριθμοι εύρεσης ελαχίστου μονοπατιού

```
initialization(G=(V,E), s)
  for every vertex v∈V
     d(v)= infty;    %distance to v
     pr(v)= nil;     %previous vertex
  d(s)=0;            %distance to source


update(u, v, w)
  if d(v) > d(u)+ w
     d(v) = d(u) + w;
     pr(v)= u
```

Αλγόριθμοι εύρεσης ελαχίστου μονοπατιού

```
Dijkstra(G=(V,E), s)
  initialize(G, s);
  S= empty;  %set of assigned vertices
  Q= V;      %set of unassigned vertices
  while Q not empty
     u= getMin(Q);  %find vertex with min cost
     S= S ∪ {u};
     for each v ∈ Adjacent(u)
          update(u, v, w);
```

Αλγόριθμοι εύρεσης ελαχίστου μονοπατιού

```
BellmanFord(G=(V,E), s)
  initialize(G, s);
     for i=1 to |V|-1
          for each edge (u,v)∈E
               update(u,v,w);
     for each edge (u,v)∈E
          if d(v)> d(u)+ w return false;
     return true;
```

Συμπίεση Αρχείων

- Υποθέστε ότι έχετε ένα αλφάβητο το οποίο αποτελείται από τους χαρακτήρες {A, B, C, D, -}.
- Υποθέστε επίσης ότι σε ένα τυχαίο κείμενο βασισμένο στο αλφάβητο αυτό οι χαρακτήρες εμφανίζονται με την εξής συχνότητα {0.35, 0.1, 0.2, 0.2, 0.15} αντίστοιχα.
- Πως μπορείτε να κωδικοποιήσετε τους χαρακτήρες έτσι ώστε να ελαχιστοποιήσετε το συνολικό μήκος του αρχείου;
  - Το ίδιο πρόβλημα εμφανίζεται επίσης στις τηλεπικοινωνίες όπου θέλετε να περιορίσετε το συνολικό μήκος του μηνύματος που θέλετε να μεταδώσετε.
- Πιθανός τρόπος επίλυσης είναι οι κώδικες σταθερού μήκους, π.χ., ASCII.

Κώδικές σταθερού μήκους.

Χαρακτήρας	A	B	C	D	-
Συχνότητα	0.35	0.1	0.2	0.2	0.15

- Εάν χρησιμοποιήσουμε κώδικες σταθερού μήκους, τότε χρειαζόμαστε τουλάχιστον
- Μια πιθανή κωδικοποίηση είναι

Χαρακτήρας	A	B	C	D	-
Κώδικας					

- Δηλαδή το συνολικό μήκος ενός αρχείου με n χαρακτήρες θα πρέπει να είναι
- Η αποκωδικοποίηση κωδικών με σταθερό μήκος είναι εύκολη. Απλά παίρνουμε 3-3 bits και ελέγχουμε το αλφάβητο.

Κώδικες μεταβλητού μήκους

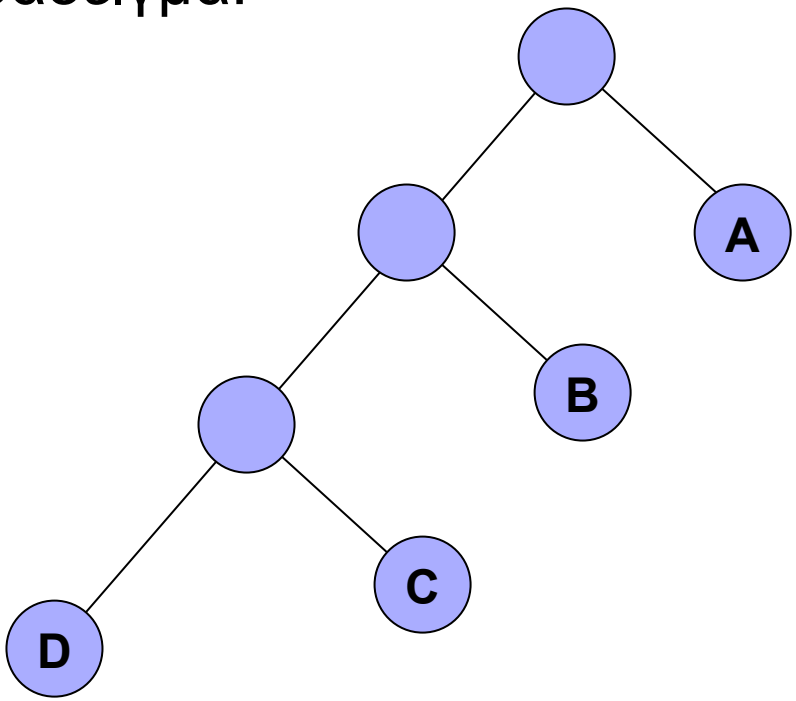
- Μια καλύτερη λύση είναι να έχουμε κώδικες μεταβλητού μήκους έτσι που οι χαρακτήρες που εμφανίζονται συχνότερα να έχουν μικρότερο μήκος από του χαρακτήρες που εμφανίζονται λιγότερο συχνά.
- Πως όμως θα μπορεί κάποιος να ξεχωρίσει τους χαρακτήρες;
  - Ο αποκωδικοποιητής διαβάζει τα bits με τη σειρά μέχρι να σχηματίσει ένα γνωστό κώδικα. Αυτό επαναλαμβάνεται μέχρι να αποκωδικοποιηθούν όλα τα δεδομένα.
  - Ο κώδικας κανενός (μικρού) χαρακτήρα δεν πρέπει να συμπίπτει με την αρχή του κώδικα κάποιου άλλου χαρακτήρα.

Κώδικες μεταβλητού μήκους

- Παράδειγμα 1:
  - Υποθέστε τους πιο κάτω κώδικες A=1, B=01, C=001, D=0001.
  - Πως αποκωδικοποιείτε το 10001101;
  -
- Παράδειγμα 2:
  - Υποθέστε ότι σε κάποιο αλφάβητο έχουμε τους πιο κάτω κώδικες A=1, B=0, C=01 και D=001.
  - Υποθέστε ότι λαμβάνεται τον κωδικό 0001. Πως το αποκωδικοποιείτε;
  -
  - Το πρόβλημα στην πιο πάνω κωδικοποίηση είναι ότι ο κώδικας του B εμφανίζεται στη αρχή του κώδικα του C και D.

Κώδικες Huffman

- Ο κώδικας κάθε στοιχείου του αλφαβήτου αντιπροσωπεύεται από ένα φύλλο ενός **δυαδικού** δέντρου στο οποίο αριστερές ακμές αντιστοιχούν σε 0 bit ενώ δεξιές ακμές αντιστοιχούν σε 1 bit.
- Παράδειγμα:



- Από το δέντρο διαβάζουμε τους κώδικες
  - A=
  - B=
  - C=
  - D=

Αλγόριθμος Huffman

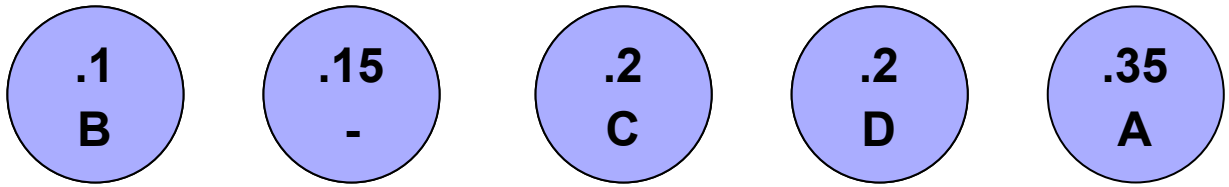
- Ο αλγόριθμος Huffman δημιουργεί ένα δέντρο (Huffman tree) στο οποίο οι πιο συχνοί χαρακτήρες έχουν μικρότερο μήκος (απόσταση από τη ρίζα) και οι λιγότερο συχνοί χαρακτήρες μεγαλύτερο μήκος.
- Από το δέντρο μπορούμε να βρούμε τους ζητούμενους κώδικες οι οποίοι ονομάζονται Huffman codes.
- Δεδομένων των συχνοτήτων κάθε χαρακτήρα και υποθέτοντας ότι η παρουσία κάθε χαρακτήρα είναι ανεξάρτητη από τη παρουσία των άλλων χαρακτήρων, ο αλγόριθμος Huffman μας δίνει βέλτιστους κώδικες (μικρότερο μέσο μήκος).

Αλγόριθμος Huffman

- Αρχικοποίηση.
  - Για κάθε χαρακτήρα δημιουργούμε ένα κόμβο στον οποίο αναγράφετε ο χαρακτήρας και η συχνότητα εμφάνισης του.
- Το επόμενο βήμα επαναλαμβάνεται n-1 φορές όπου n είναι ο αριθμός των χαρακτήρων στο αλφάβητο.
  - Βρίσκουμε τους δύο κόμβους με τη μικρότερη συχνότητα
  - Τους κάνουμε παιδιά ενός νέου κόμβου στο οποίο η συχνότητα είναι το άθροισμα των συχνοτήτων των παιδιών.

Παράδειγμα Αλγόριθμου Huffman

Χαρακτήρας	A	B	C	D	-
Συχνότητα	0.35	0.1	0.2	0.2	0.15



Παράδειγμα Αλγόριθμου Huffman

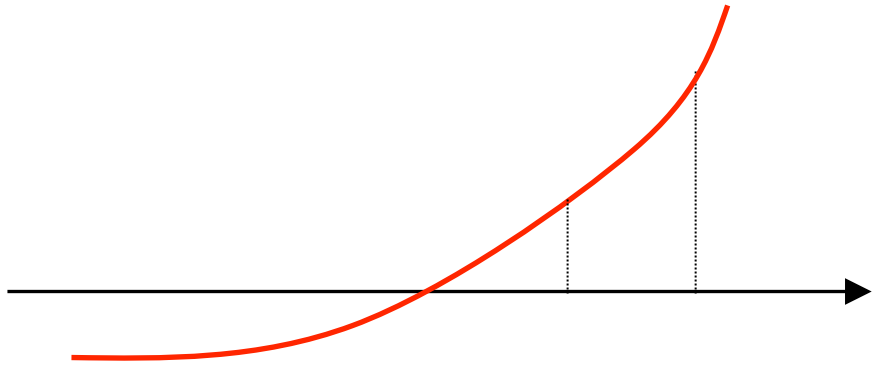
Παράδειγμα Αλγόριθμου Huffman

- Από το δέντρο διαβάζουμε τους κώδικες
  - A=
  - B=
  - C=
  - D=
  - -=

- Μέσο μήκος κώδικα ανά χαρακτήρα:

Επίλυση μη γραμμικών εξισώσεων μιας μεταβλητής

- Πώς μπορείτε να βρείτε τη λύση της μη γραμμικής εξίσωσης $f(x)=0$ ξεκινώντας από ένα αρχικό σημείο x^0.



- Γενικός Αλγόριθμος