BIO 285/CSCI 285/MATH 285
Bioinformatics
Programming Lecture 9
Sequence Alignment 3
Dynamic Programming

Instructor: Lei Qian
Fisk University

Alignment

Calculate scores of an alignment:

```
def score(s1, s2):
    sc = 0
    for i in range(len(s1)):
        if s1[i]==s2[i]:
                sc+= 20
        elif (s1[i]=='A' and s2[i]=='G') or \
        (s1[i]=='G' and s2[i]=='A'):
                sc+=10
        #.....
    return sc

strand1 = "ACTCCG"
strand2 = "CGACGC"
print score(strand1, strand2)
```

Alignment

Calculate scores of an alignment – using dictionaries:

```
scoreTable={"AA":20,"AC":5,"AG":10,"AT":5, \
            "CA":5,"CG":5,"CC":20,"CT":5, \
            "GA":10, "GC":5,"GG":20, "GT": 5, \
            "TA":5, "TC":10,"TG":5, "TT": 20}
def score(s1, s2):
    sc = 0
    for i in range(len(s1)):
        pair = s1[i]+s2[i]
        sc+=scoreTable[pair]
    return sc

strand1 = "ACTCCG"
strand2 = "CGACGC"
print score(strand1, strand2)
```

Alignment

Another scoring method:

Match:	**1**
Mismatch:	**-1**
Gap:	**-4**
Extend Gap:	**-1**

```
ACTCT-CTGCCCTGCTGG
 CTATAC---CCTGCTGGG
 MMSMGMGEEMMMMMMMMM

M: Match
S: Mismatch
G: Gap
E: Extend
11 Matches, 1 mismatch,2 gaps, 2 extended gaps
Score = 12-1-2*4-2 = 1.
```

Alignment

Alignment Games:

UIUC TechEnG Alignment Game:

1. Sequence Alignment Game
http://teacheng.illinois.edu/SequenceAlignment/

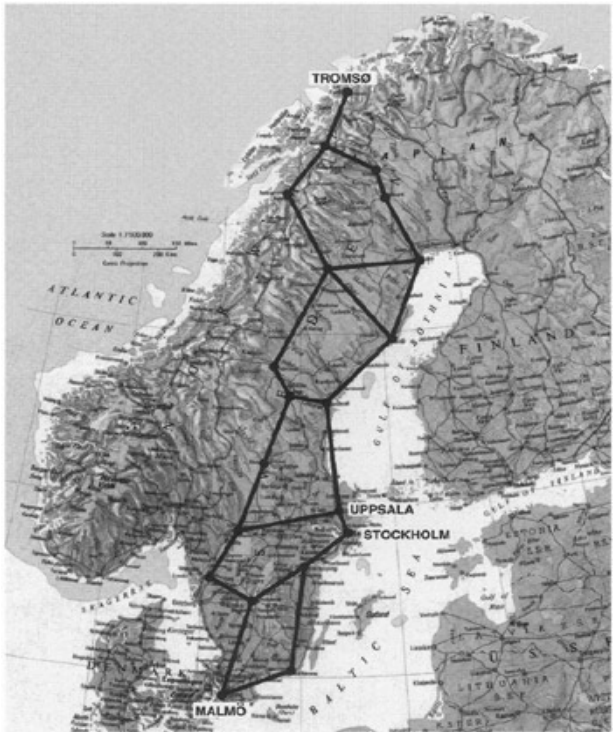
2. Dynamic Programming Game
http://teacheng.illinois.edu/SequenceAlignmentDP/

Phylo
http://phylo.cs.mcgill.ca/

Alignment

The dynamic programming: solving complex problems by breaking them down into simpler subproblems

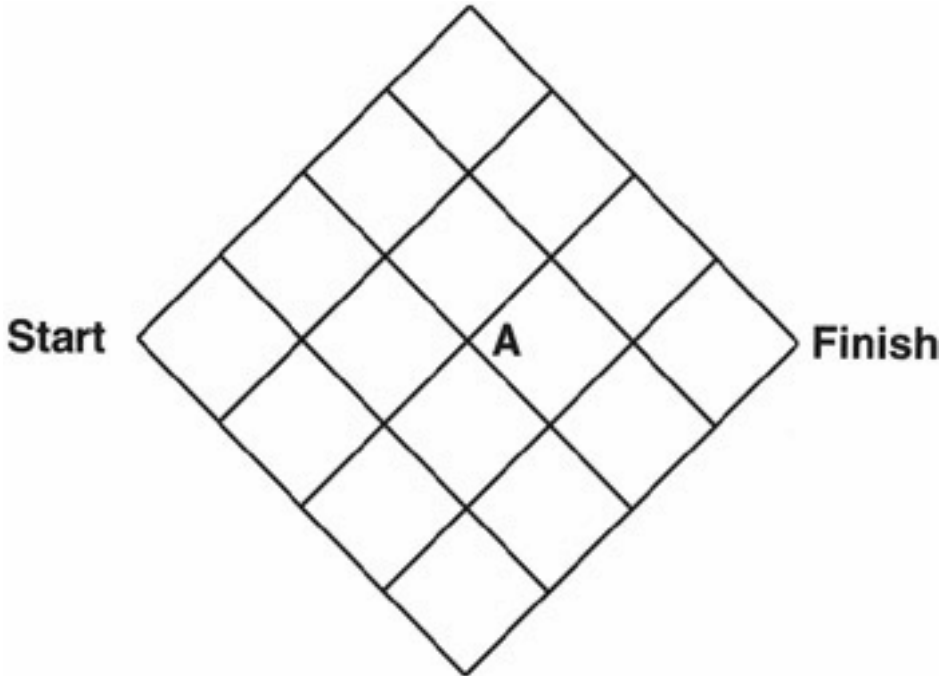
Example: Find the shortest path in a graph.



Alignment

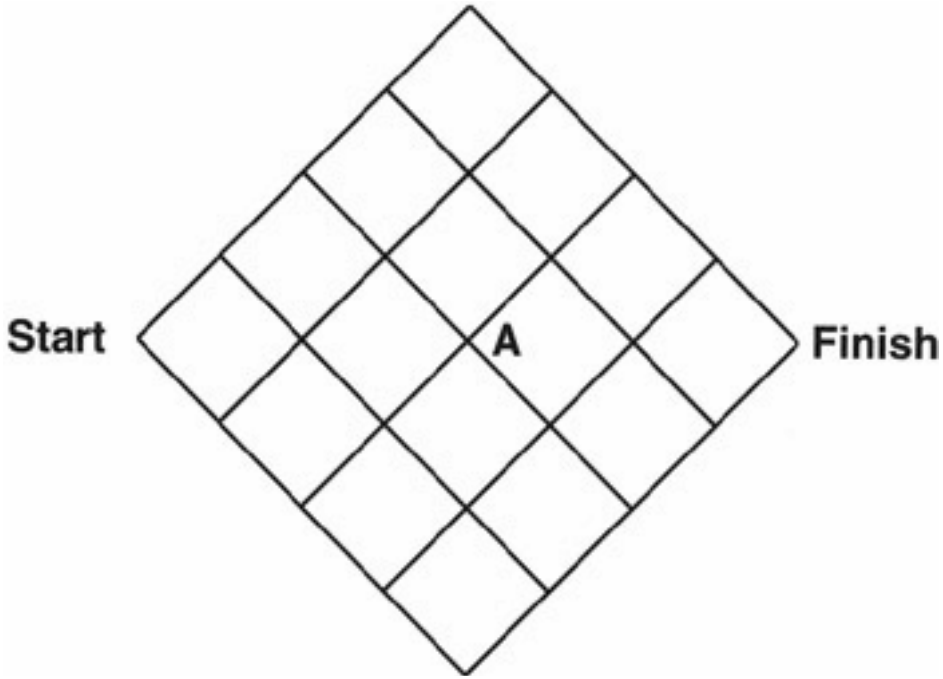
The dynamic programming

The shortest path from Start to Finish through A must consists of the shortest path from Start to A and the shortest path from A to Finish.

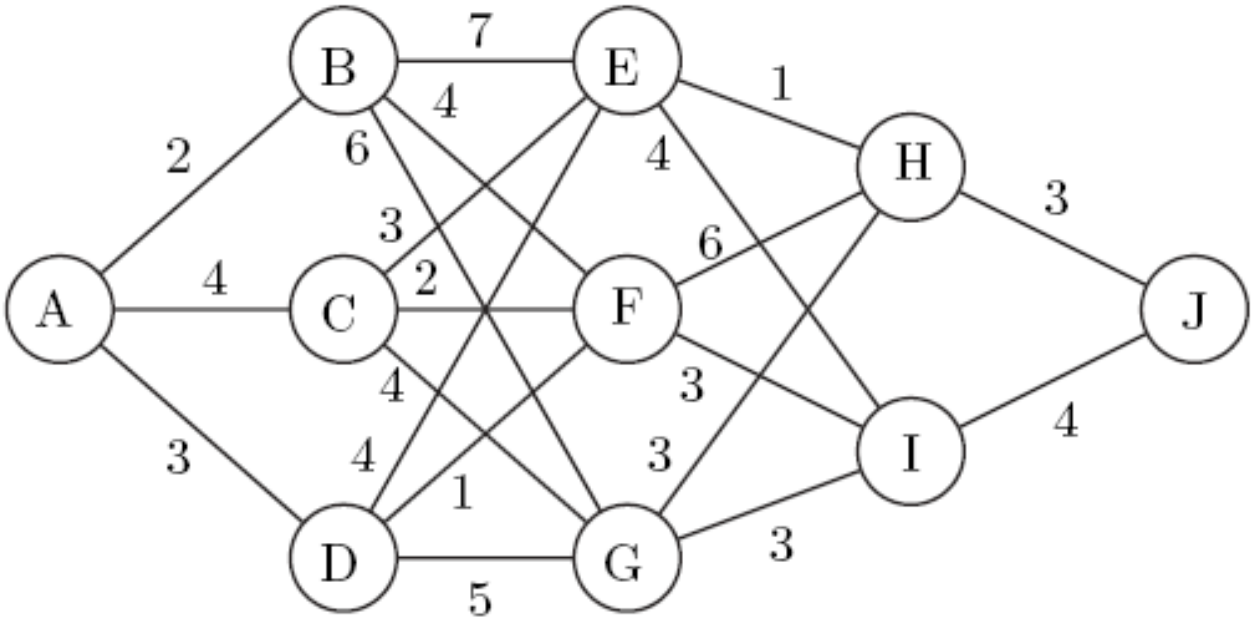


Alignment

Dynamic programming :
An abstract version of the problem:
There are 6 paths from start to A and 6 paths from A to finish.
There are 36 total possible path from start to finish through A.

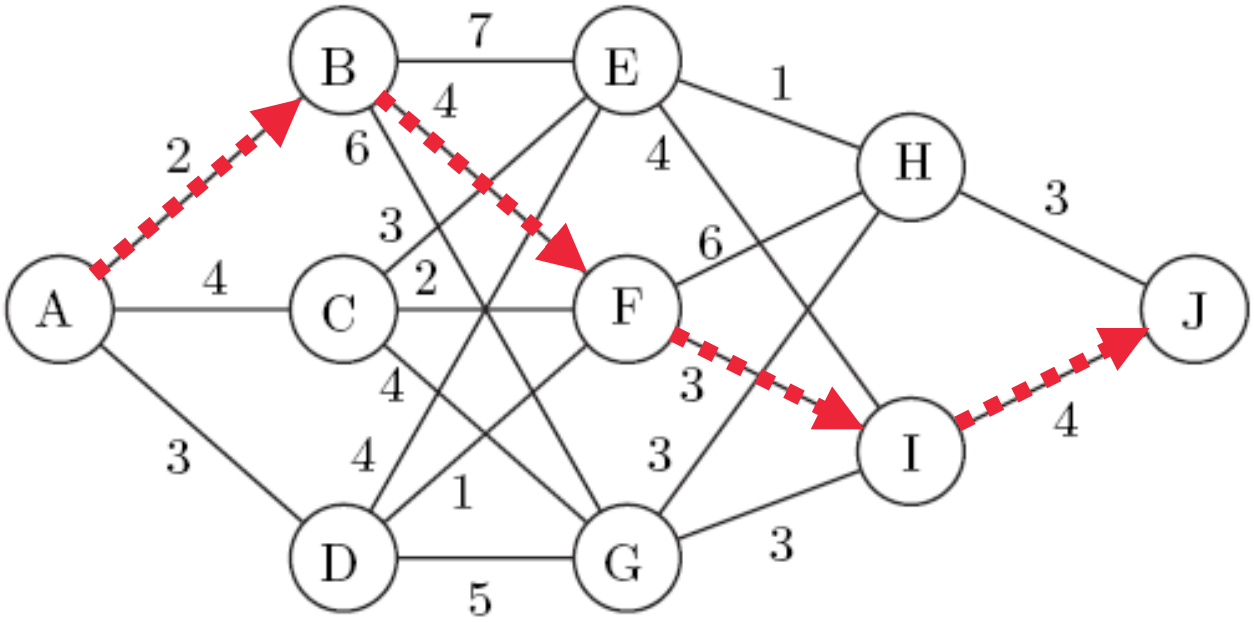


Stagecoach problem



A traveler wishes to minimize the length of a journey from town **A** to **J.**

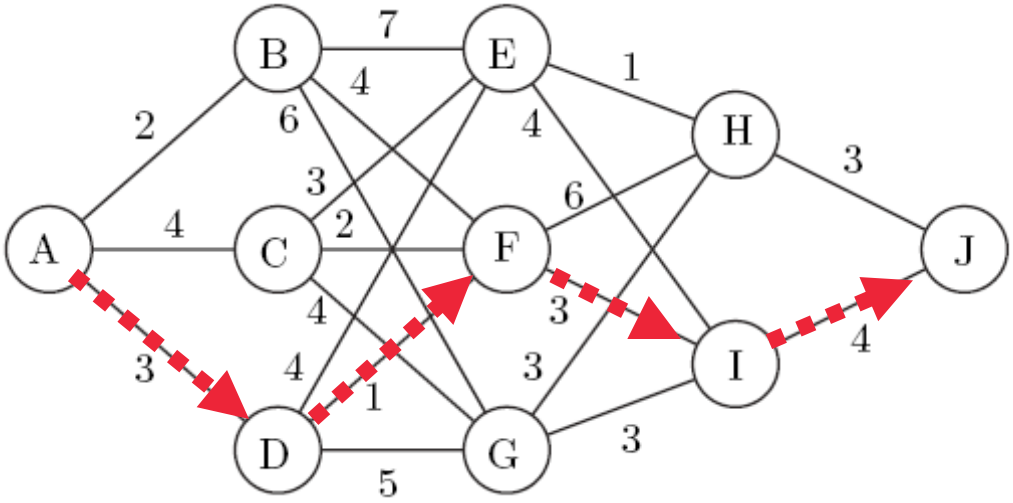
Greedy algorithm



The length of the route A-B-F-I-J: 2+4+3+4=13.

Can we find shorter route?

Exhaustive search: try all

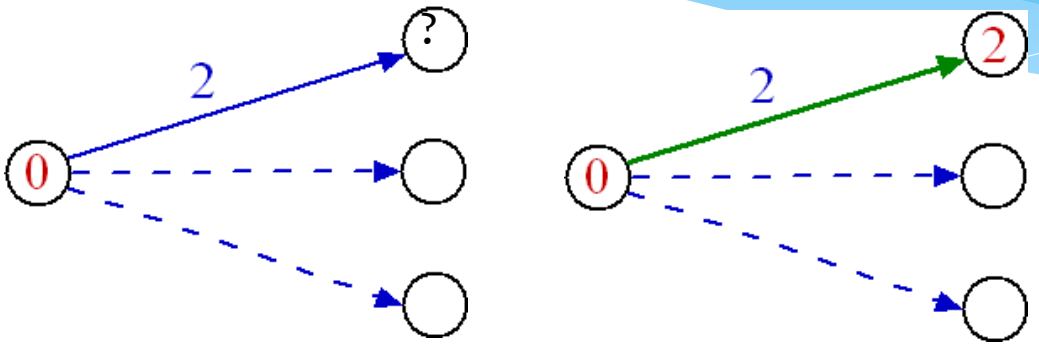


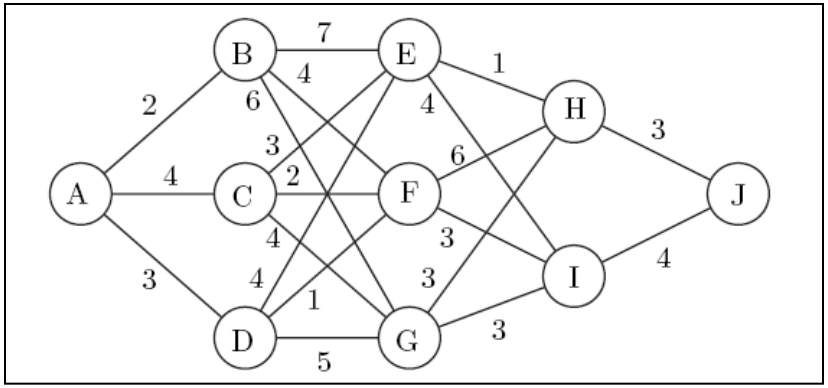
Route A-D-F-I-J: 3+1+3+4=11

The total number of routes to be tested: 3×3×2×1=18

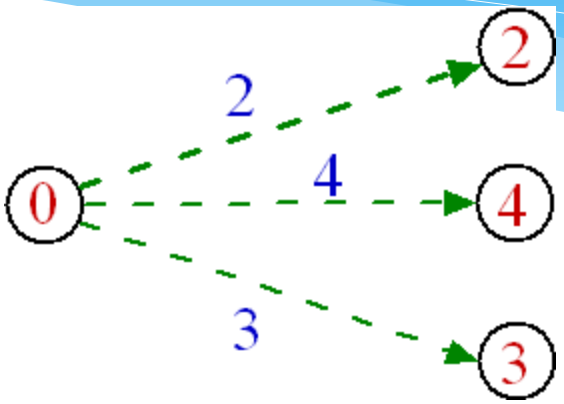
Can we avoid exhaustive search?

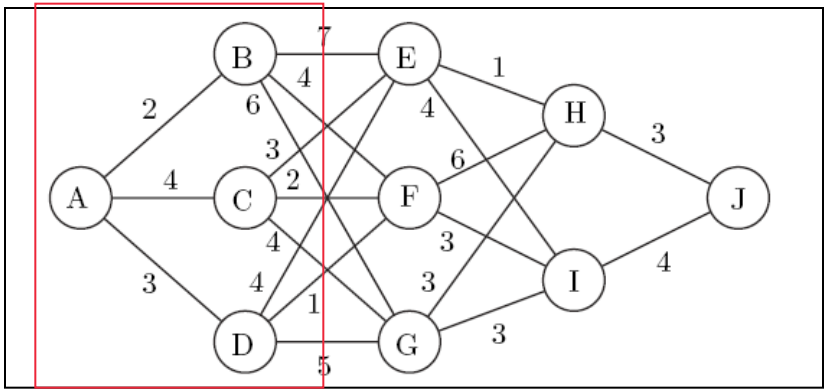
Shortest path construction: 1st stage (B)





Shortest path construction: 1st stage



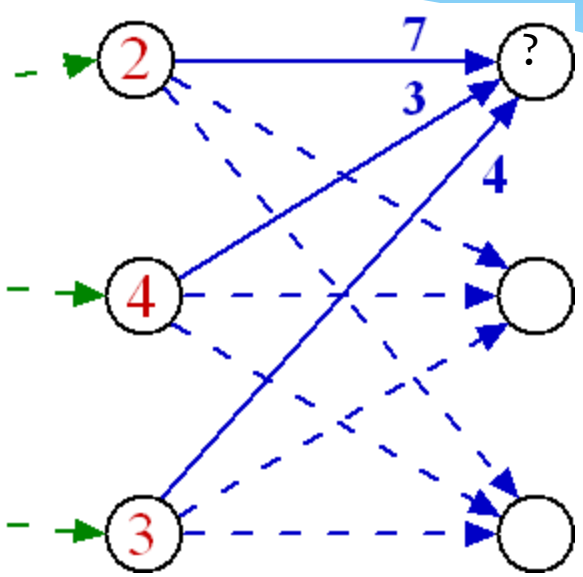


S(A,B)=2

S(A,C)=4

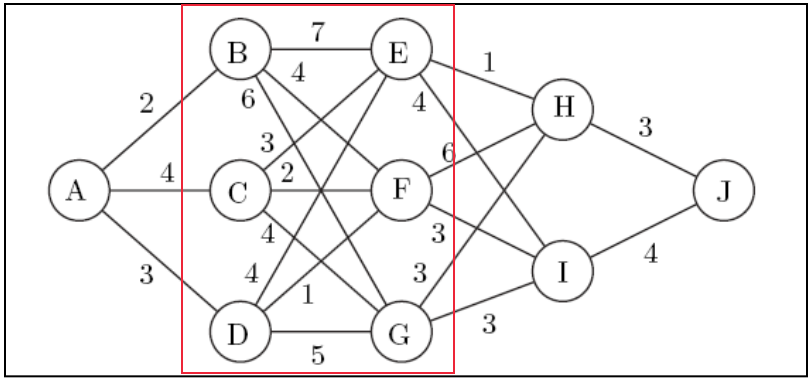
S(A,D)=3

Shortest path construction: 2nd stage (E)

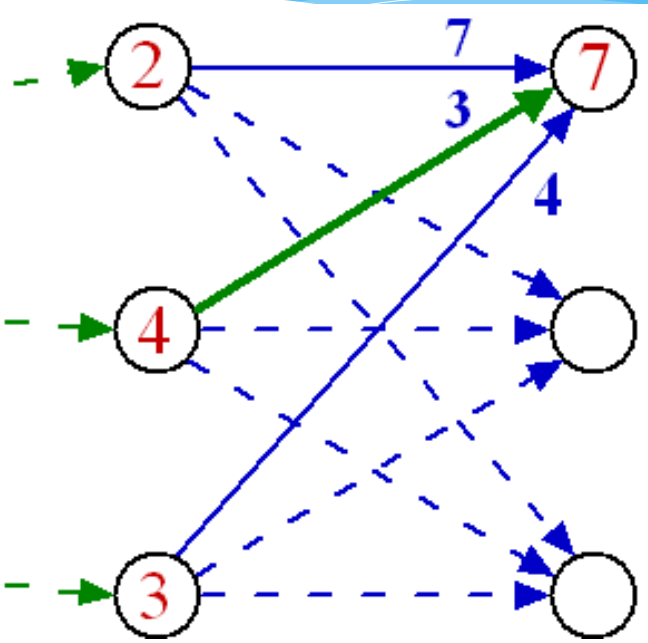


1. (A..B)-E: 2+7=9
2. (A..C)-E: 4+3=7
3. (A..D)-E: 3+4=7

(A..C)-E: 7

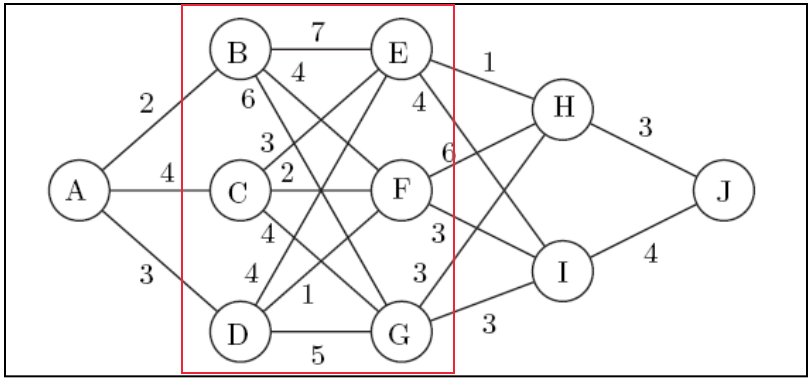


Shortest path construction: 2nd stage (E)

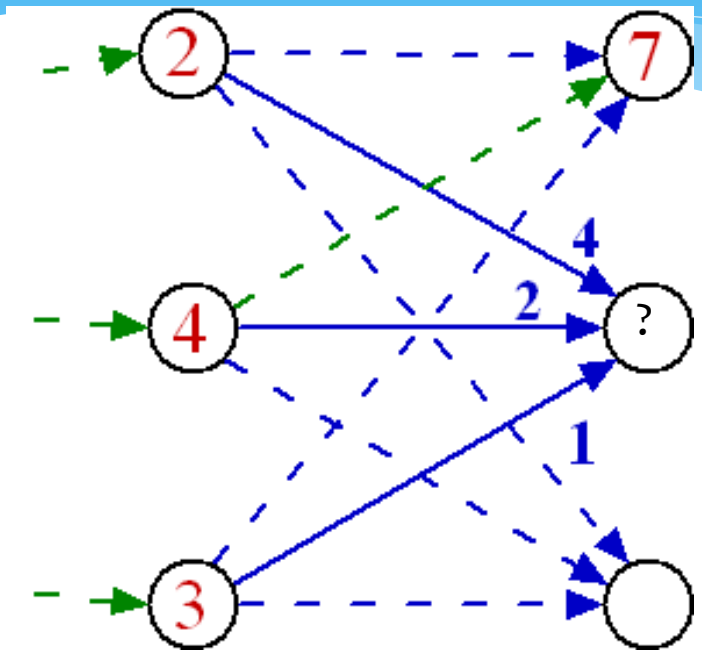


1. (A..B)-E: 2+7=9
2. (A..C)-E: 4+3=7 *)
3. (A..D)-E: 3+4=7

(A..C)-E: 7

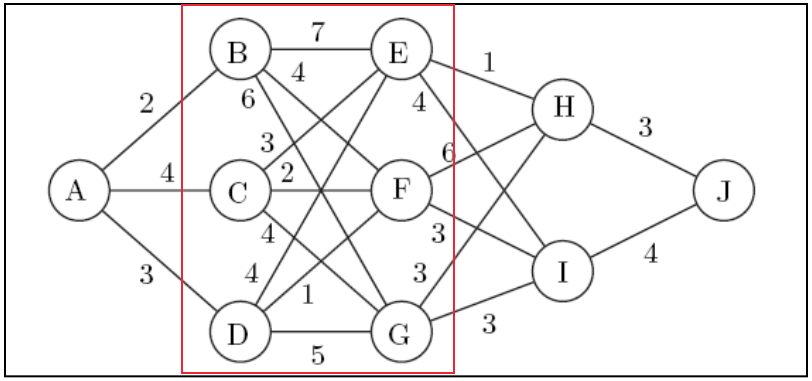


Shortest path construction: 2nd stage (F)

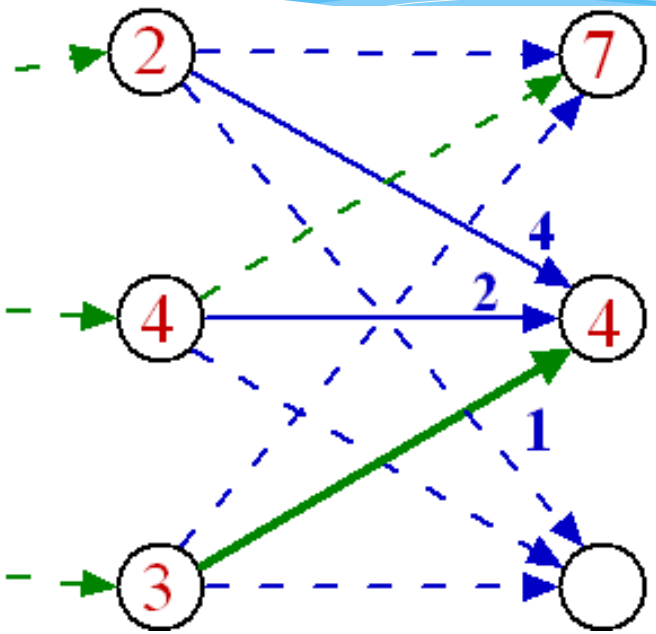


1. (A..B)-F: 2+4=6
2. (A..C)-F: 4+2=6
3. (A..D)-F: 3+1=4 *)

(A..C)-F: 4

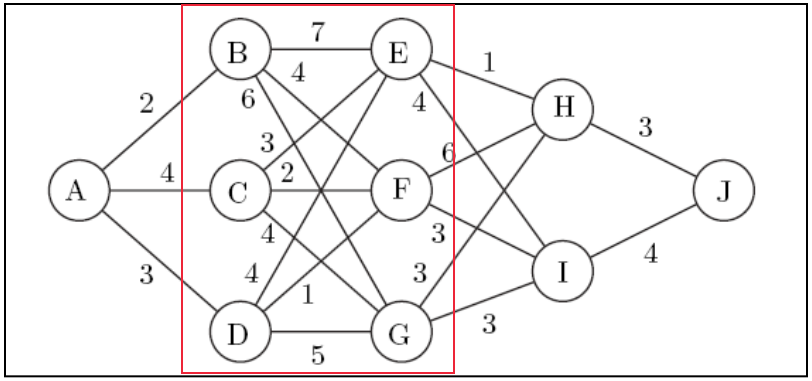


Shortest path construction: 2nd stage (F)

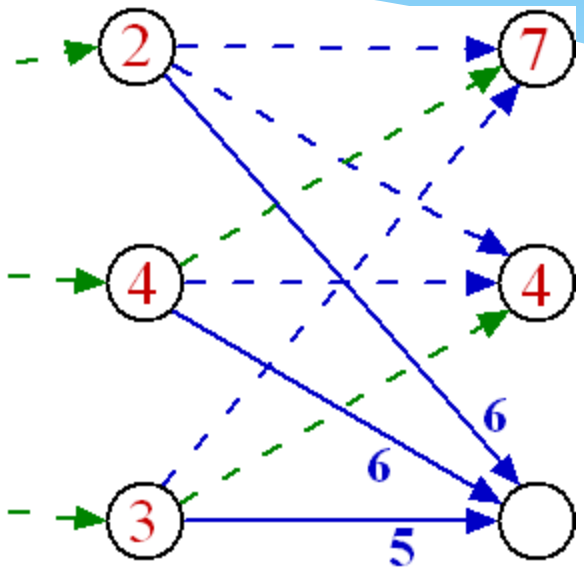


1. (A..B)-F: 2+4=6
2. (A..C)-F: 4+2=6
3. (A..D)-F: 3+1=4 *)

(A..D)-F: 4

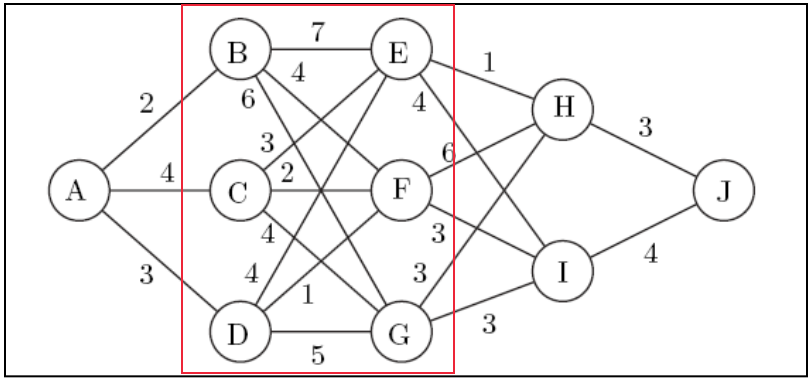


Shortest path construction: 2nd stage (G)

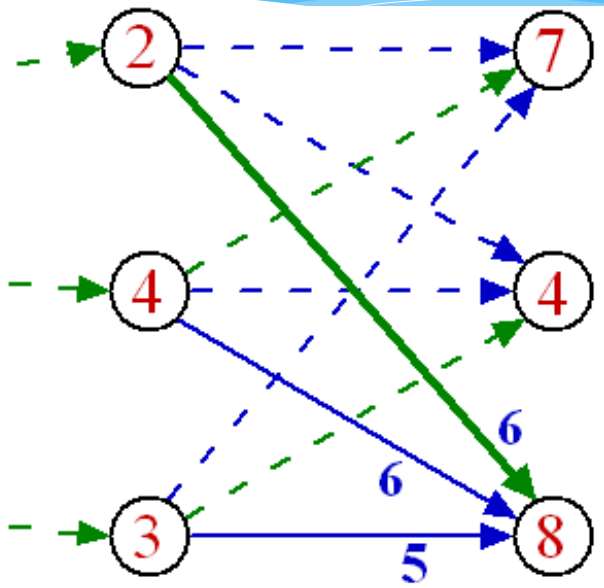


1. (A..B)-G: 2+6=8 *)
2. (A..C)-G: 4+6=10
3. (A..D)-G: 3+5=8

(A..B)-G: 8

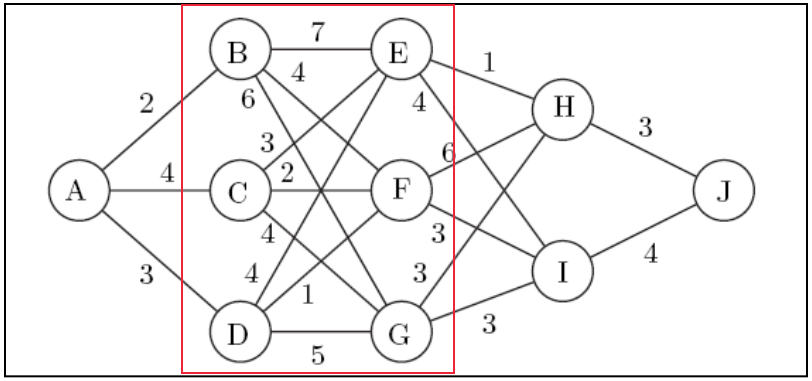


Shortest path construction: 2nd stage (G)

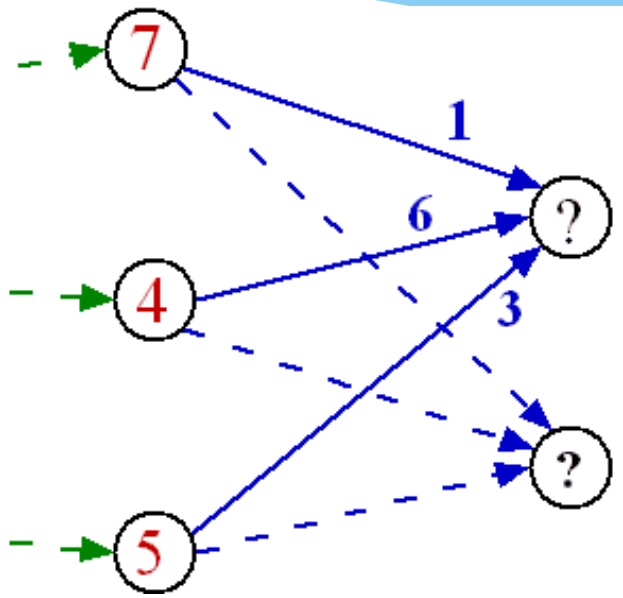


1. (A..B)-G: 2+6=8 *)
2. (A..C)-G: 4+6=10
3. (A..D)-G: 3+5=8

(A..B)-G: 8

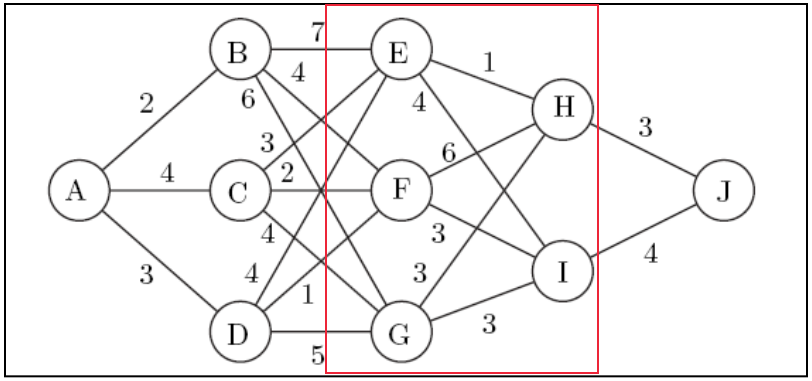


Shortest path construction: 3rd stage (H)

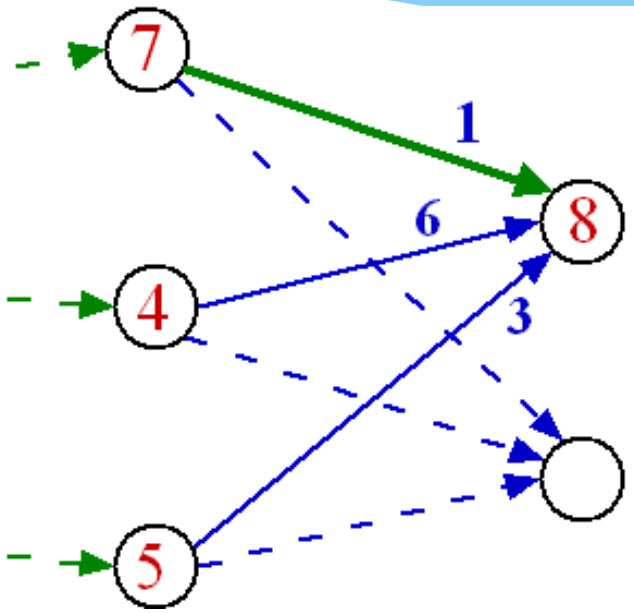


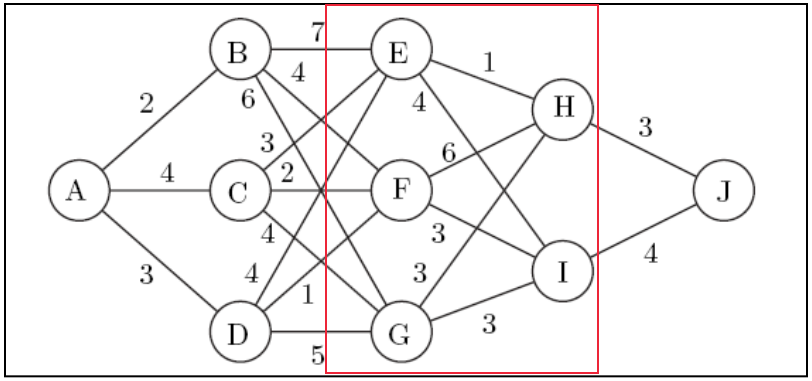
1. (A..E)-H: 7+1=8 *)
2. (A..F)-H: 4+6=10
3. (A..G)-H: 5+3=8

(A..E)-H: 5



Shortest path construction: 3rd stage (H)

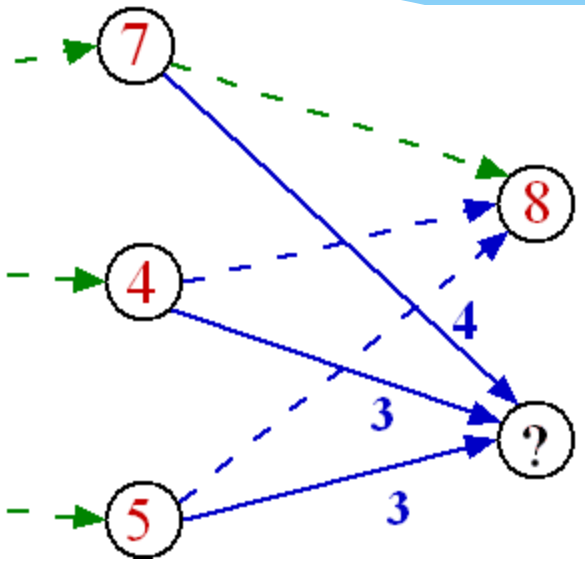




1. (A..E)-H: 7+1=8 *)
2. (A..F)-H: 4+6=10
3. (A..G)-H: 5+3=8

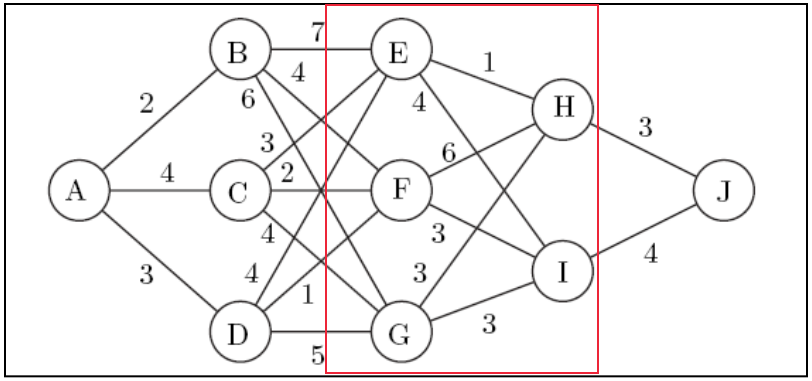
(A..E)-H: 5

Shortest path construction: 3rd stage (I)

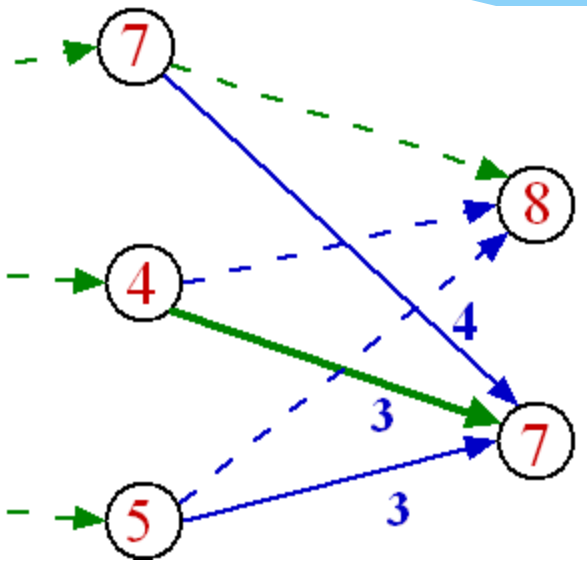


1. (A..E)-I: 7+4=11
2. (A..F)-I: 4+3=7 *)
3. (A..G)-I: 5+3=8

(A..F)-I: 7

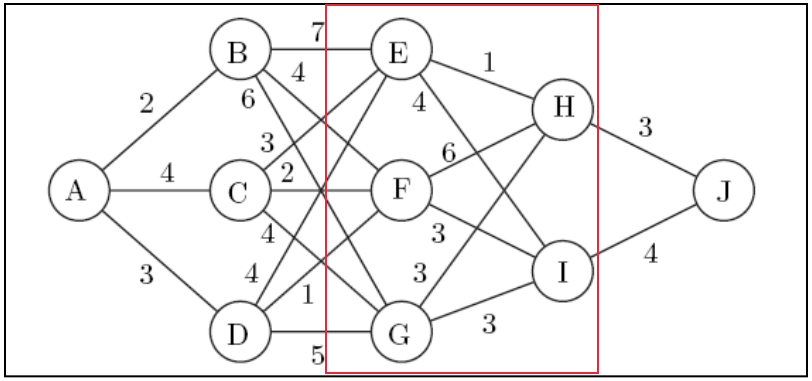


Shortest path construction: 3rd stage (I)

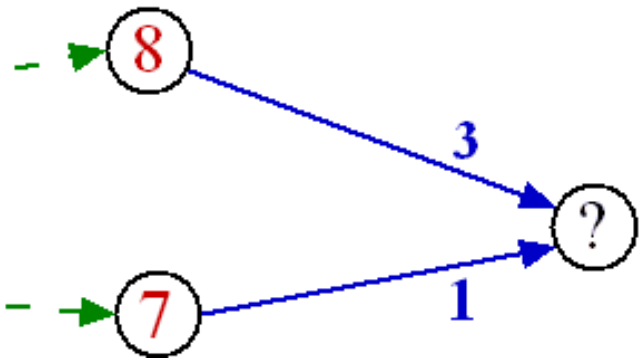


1. (A..E)-I: 7+4=11
2. (A..F)-I: 4+3=7 *)
3. (A..G)-I: 5+3=8

(A..F)-I: 7

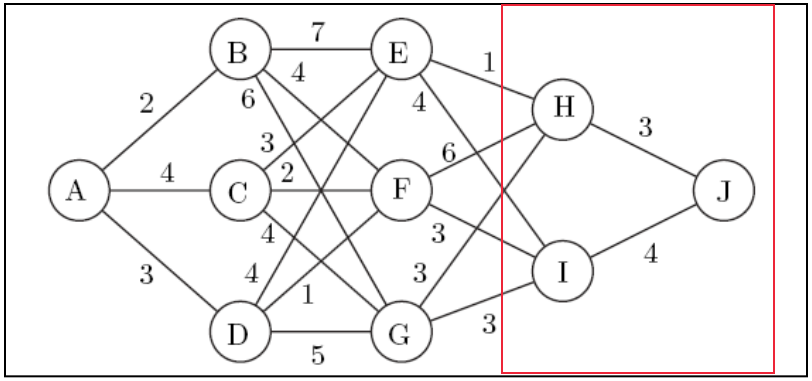


Shortest path construction: 4th stage (J)

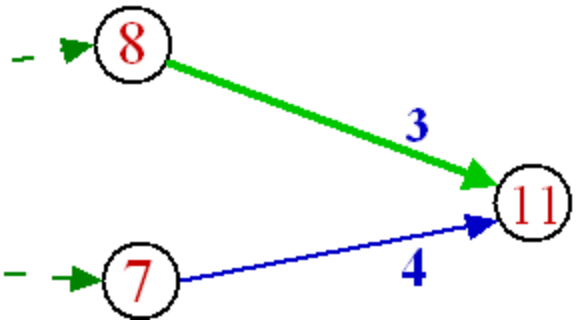


1. (A..H)-J: 8+3=11 *)
2. (A..I) -J: 7+4=11

(A..H)-J: 11

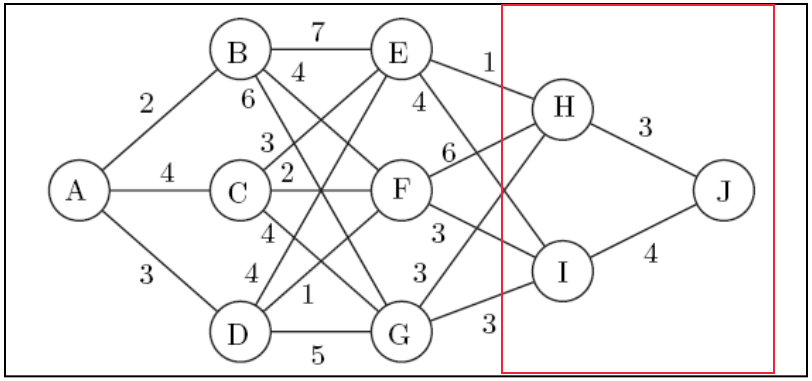


Shortest path construction: 4th stage (J)

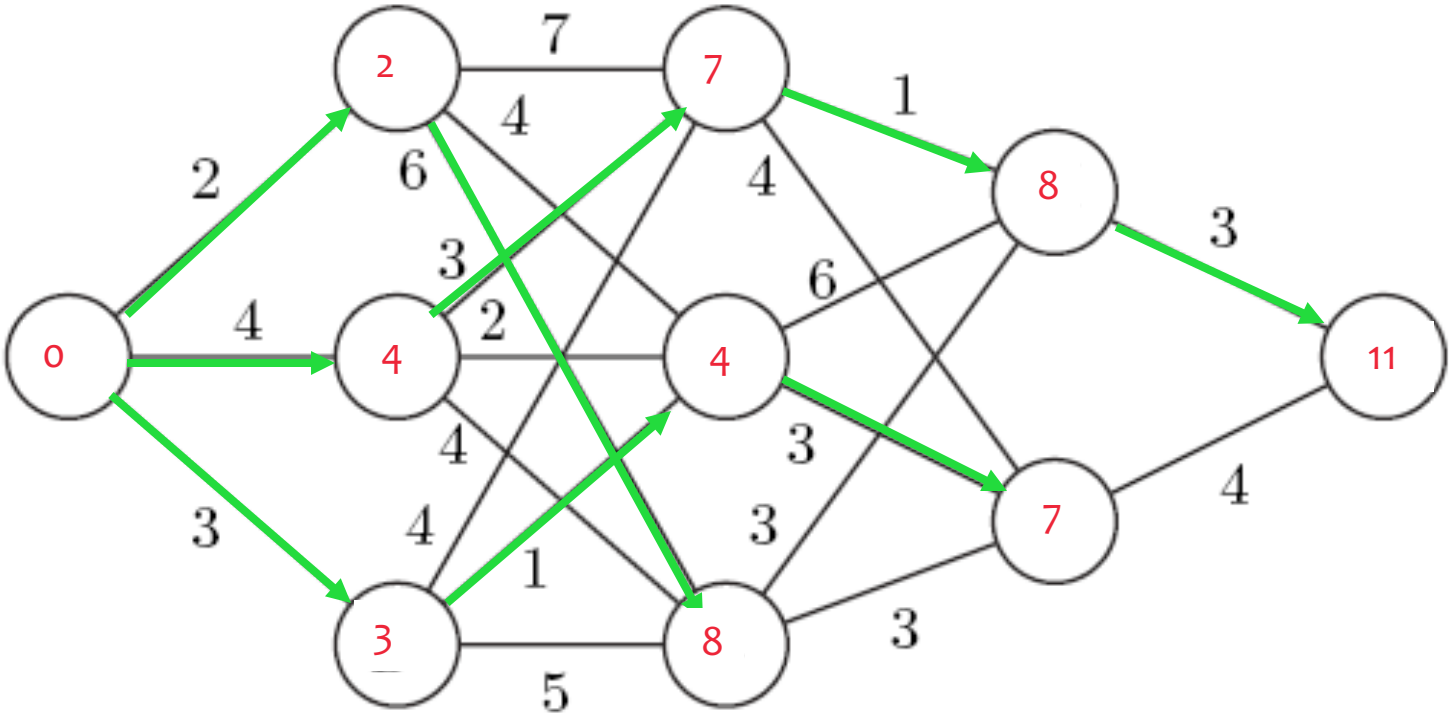


1. (A..H)-J: 8+3=11 *)
2. (A..I) -J: 7+4=11

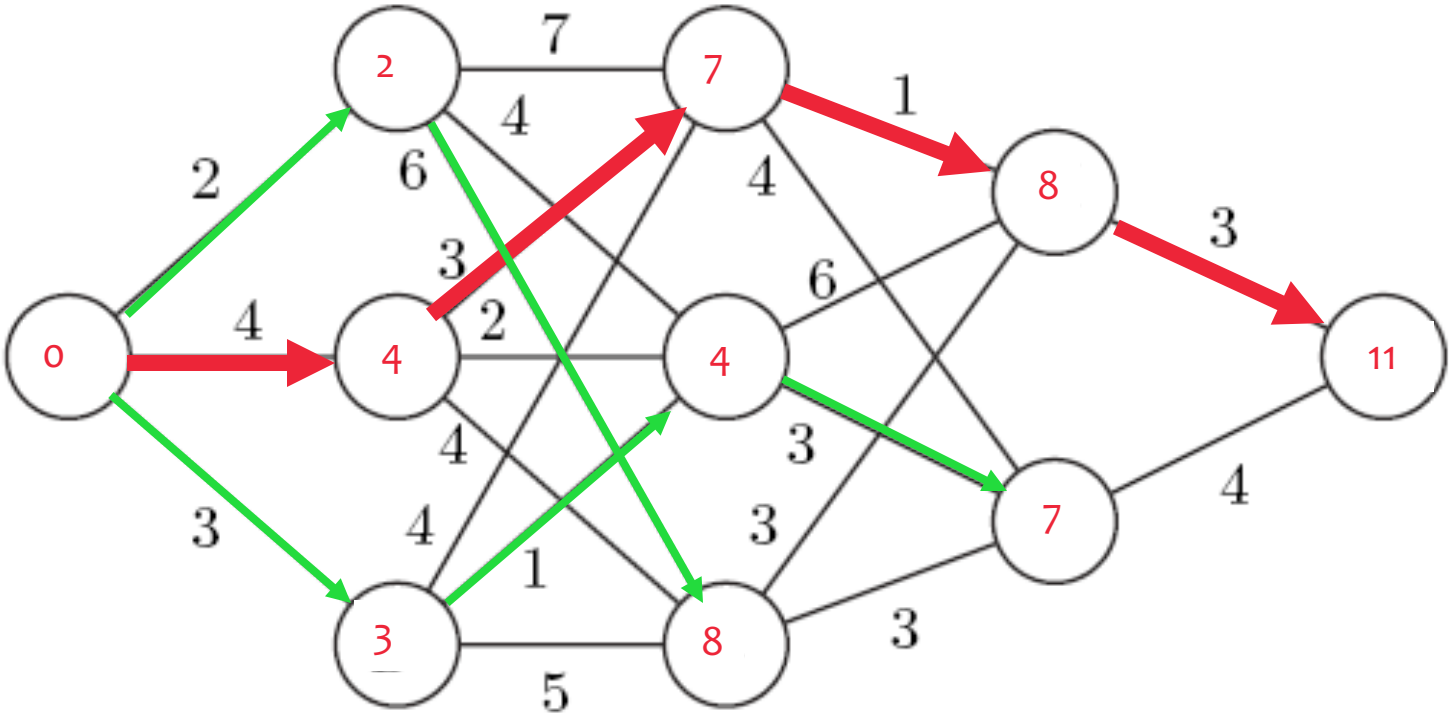
(A..H)-J: 11



Backtrack the shortest path



The shortest path



Route A-C-E-H-J: 4+3+1+3=11