**Homework – 吴伟伟**

**Q1.** Prove that *n*5 = *O*(2*n*).

Proof:

Assume

*T(n)= n5, f(n)= 2n*

For bit-O annotation, we need to find C>0 and N>0 that

*T(n) ≤C·f(n), ∀n>N.*

Let C=1, N=23, then

*n5≤1·2n , ∀ n≥23.*

So

*n5 = O(2n).*

**Q2.** Prove that 100*n* + log*n* = *O*(*n* + (log*n*)2).

Proof:

Assume

*T(n)= 100n + logn, f(n)= n + (logn)2*

Let C=100, N=3, then

*Cf(n)-T(n)= 100(logn)2-logn*

We have (logn)2≥logn≥loge=1 when n≥3>e, then

*100(logn)2-logn > 0, ∀ n≥3.*

Means

*100f(n) ≥T(n), ∀ n≥3.*

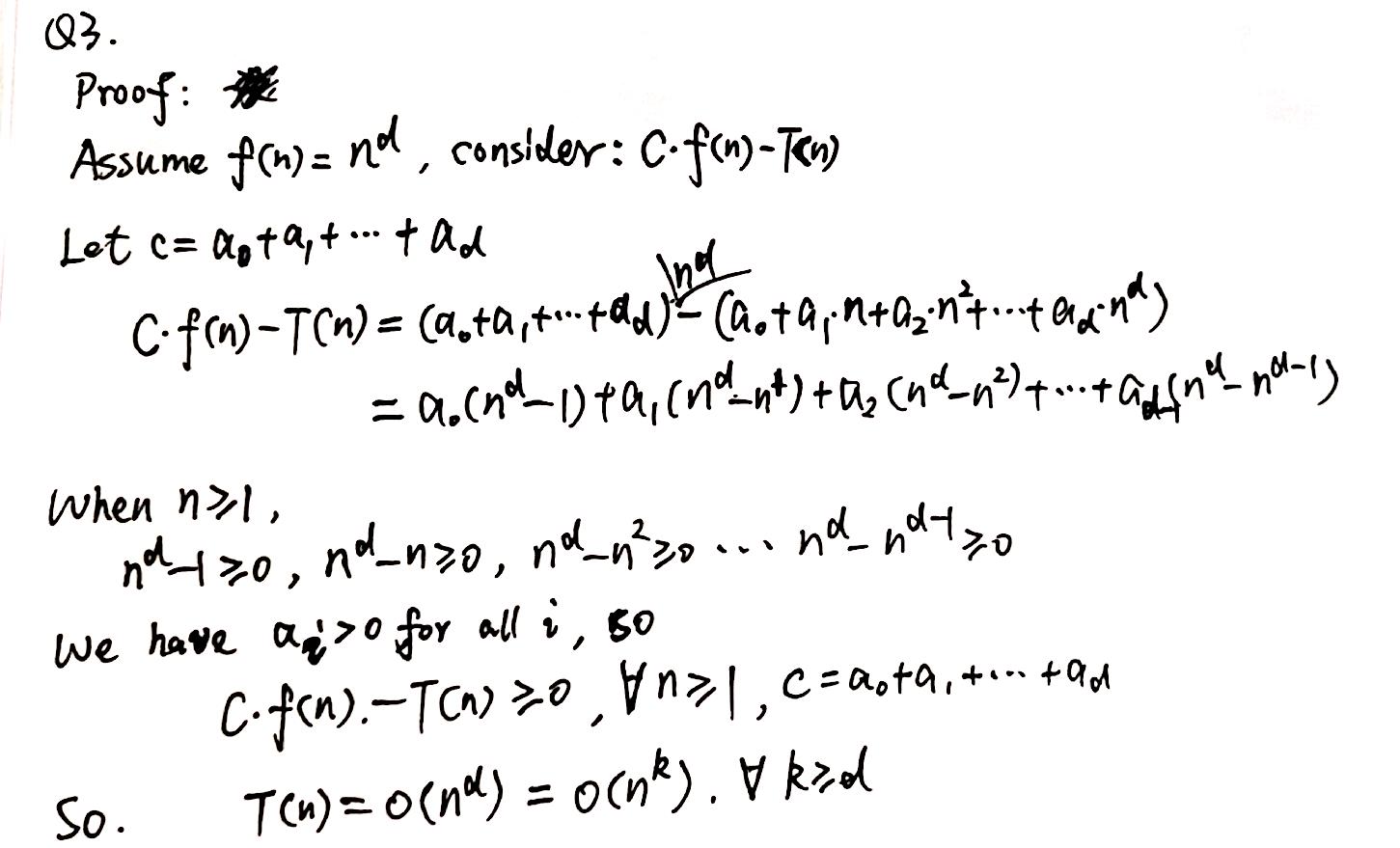
So

*100n + logn = O(n + (logn)2)*.

**Q3.** Consider, with constants *d >* 0 and *ai >* 0 for all *i*. Prove that for any *k* ≥ *d*, it holds that

*T*(*n*) = *O*(*nk*)*.*

Answer (hand writing is simpler for this question):

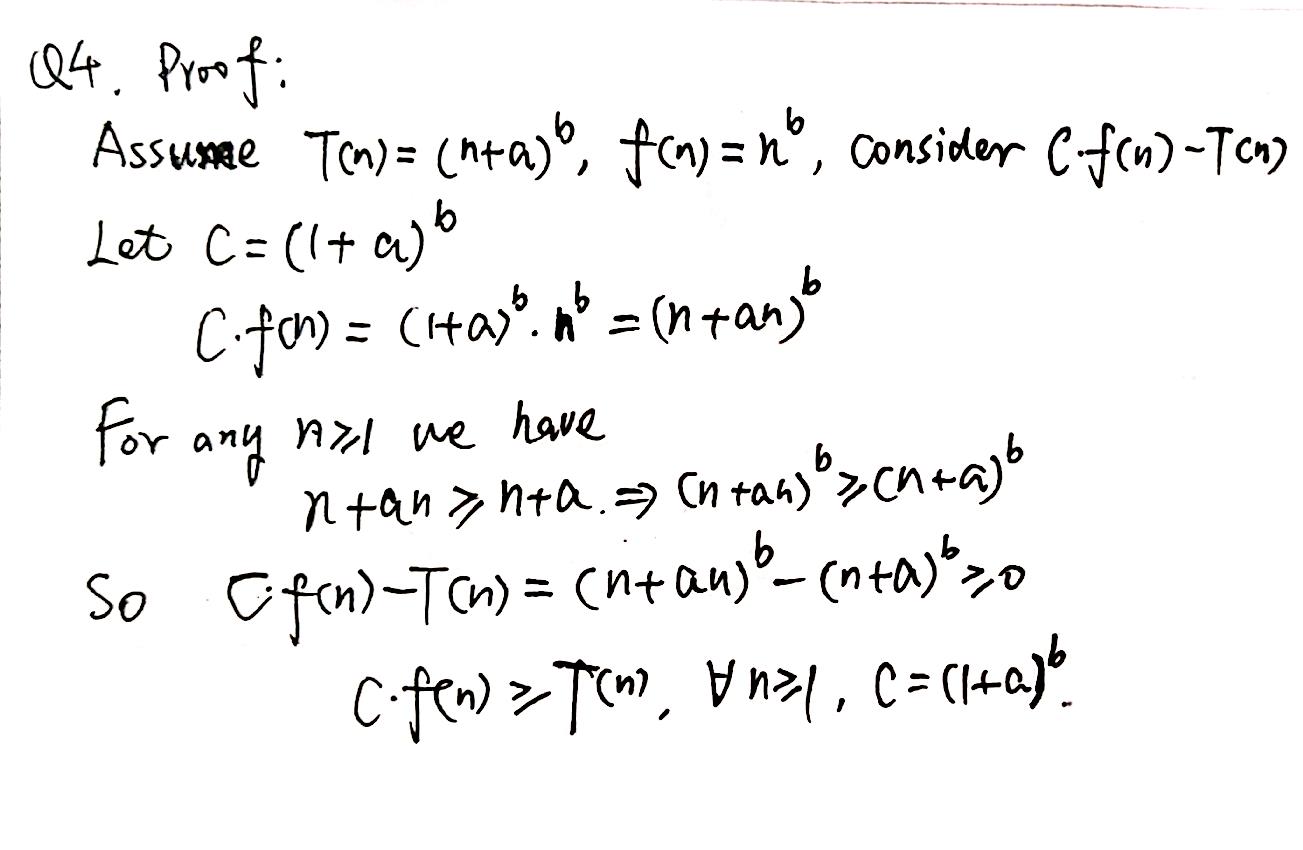


**Q4.** Show that for any real constant *a >* 0 and *b >* 0, it holds that

(*n* + *a*)*b* = *O*(*nb*)*.*

You can assume that *a* and *b* are all integers.

Answer (hand writing is simpler for this question):



**Q5.** Describe the details of applying counting-sort with the following numbers,

1*,*5*,*4*,*7*,*2*,*2*,*1*,*1*,*4*,*3*,*1*,*2*,*9

which are selected from {0*,*1*,*2*,*··· *,*9}.

Answer:

a=[1,5,4,7,2,2,1,1,4,3,1,2,9]

counting for 0-9 in a and stored in c with 0-9 indexes

c=[0,4,3,1,2,1,0,1,0,1]

sorted positions array can be accumulated by c[i]=c[i]+c[i-1] for i from 1 to 9

c=[0,4,7,8,10,11,11,12,12,13]

find positions in c and decrease by 1 for each element in a, sorted numbers are stored into s

s=[0,0,0,1,0,0,0,0,0,0,0,0,0]

s=[0,0,0,1,0,0,0,0,0,0,5,0,0]

s=[0,0,0,1,0,0,0,0,0,4,5,0,0]

s=[0,0,0,1,0,0,0,0,0,4,5,7,0]

s=[0,0,0,1,0,0,2,0,0,4,5,7,0]

s=[0,0,0,1,0,2,2,0,0,4,5,7,0]

s=[0,0,1,1,0,2,2,0,0,4,5,7,0]

s=[0,1,1,1,0,2,2,0,0,4,5,7,0]

s=[0,1,1,1,0,2,2,0,4,4,5,7,0]

s=[0,1,1,1,0,2,2,3,4,4,5,7,0]

s=[1,1,1,1,0,2,2,3,4,4,5,7,0]

s=[1,1,1,1,0,2,2,3,4,4,5,7,0]

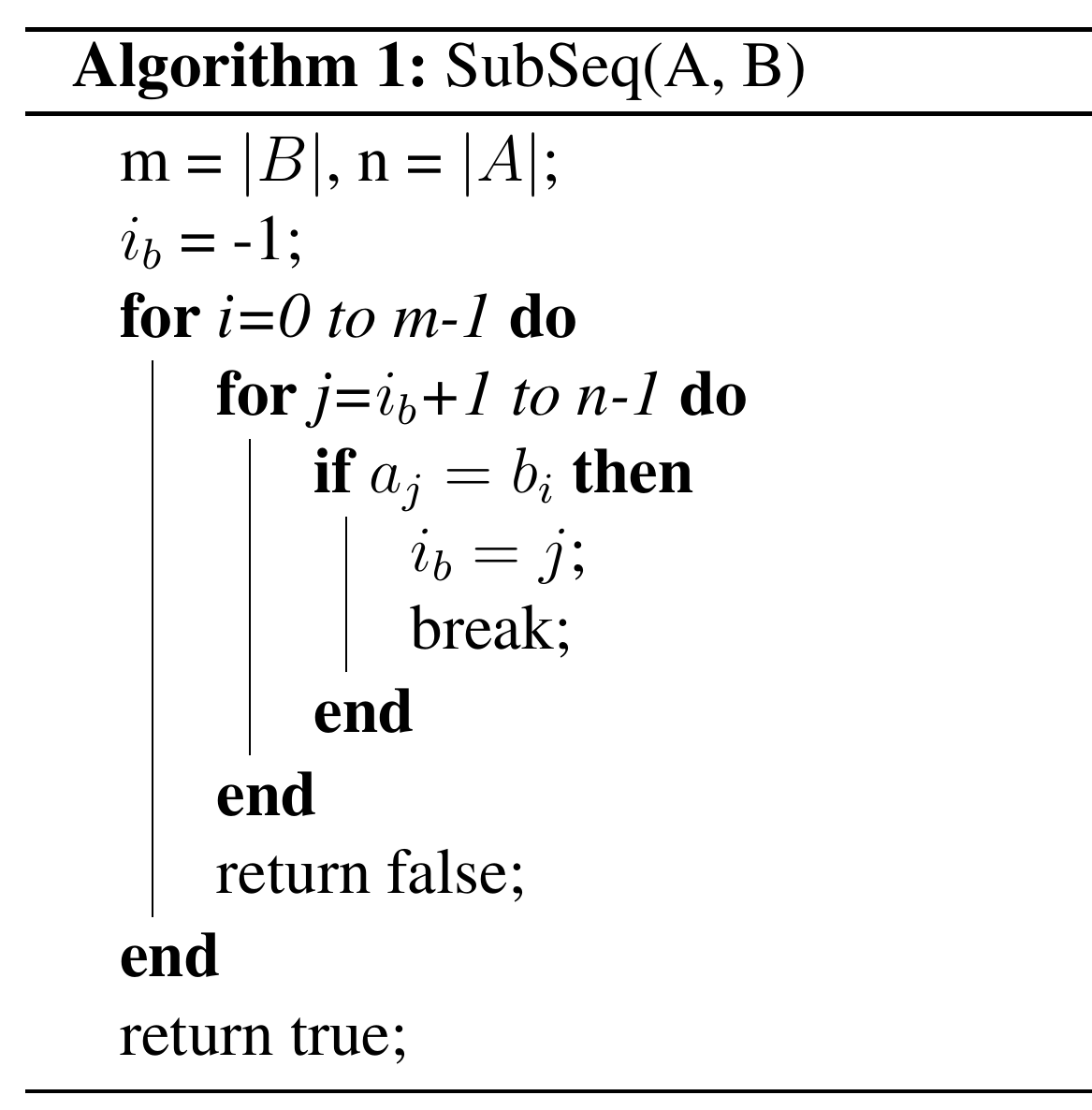
s=[1,1,1,1,2,2,2,3,4,4,5,7,0]

s=[1,1,1,1,2,2,2,3,4,4,5,7,9]

**Q6.** Recall that given a sequence *A* = *a*0*a*1 ···*an*−1 of size *n*, a subsequence has the form of *ai*0*ai*1 ···*aik*−1 with *i*0 *< i*1 *<* ··· *< ii*−1. Given a string *B* = *b*0*b*1 ···*bm*−1 with *m < n*, describe an algorithm to determine if *B* is a subsequence of *A*. Analyze the complexity of your algorithm in the big-*O* notation.

Answer:

Pseudocode description:



**Complexity: O(m·n)**

1st tier for loop: m times

2nd tier for: n at most

So the time complexity is O(m·n).

**Q7.** *T*(*n*) = 10 · *T*(*n/*3) + *n*2*.*5. Determine the order of *T*(*n*), in the big-Θ notation.

Answer: Consider T(n) = a· T(n/b) + O(nd)

a=10, b=3, d=2.5

We get

logba=log310>d=2.5

According to the Master Theorem, when logba>d

T(n) = Θ(nlogba) = Θ(nlog310)

**Q8.** Let *F*(*n*) be the number of “hello” printed by algorithm *f*(*n*). For example, *F*(0) = *F*(1) = 0, *F*(2) = 1.

**Algorithm1:**

*f*

(

*n*

)

**if**

*n>*

1

**then**

print“Hello”;

*f*

(

*n/*

2)

;

*f*

(

*n/*

2)

;

Determine the order of *F*(*n*) in the big-Θ notation

Answer:

From the algorithm pseudocode, we can get

T(n)=2T(n/2)+O(1)

Means

a=2, b=2, d=0

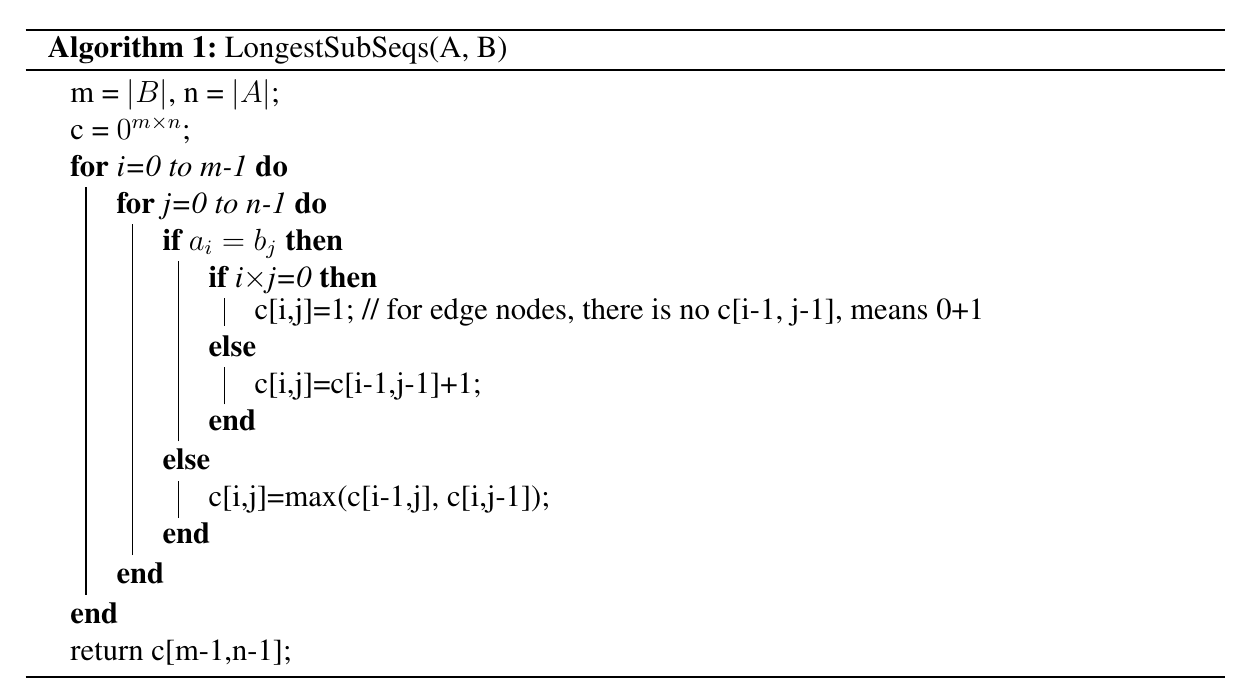
logba=1>d=0

According to the Master Theorem, when logba>d

T(n) = Θ(nlogba) = Θ(n).

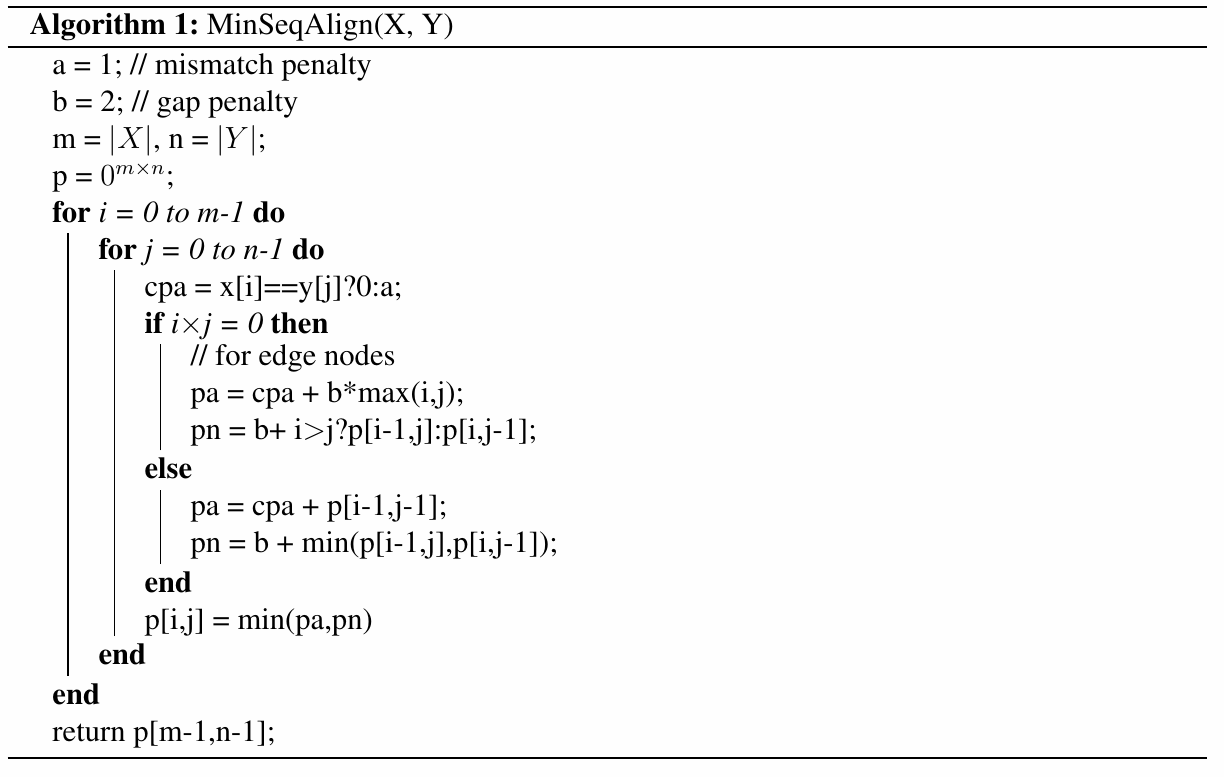
**Q9.** Give the pseudocode of the dynamic programming algorithm for solving the longest common subsequence problem.

Answer:



**Q10.** Give the pseudocode of the dynamic programming algorithm for solving the sequence alignment problem.

Answer:



**Q11.** Calculate the length of the shortest paths from *s* to *t*, using Dijkstra’s algorithm (for the directed graph) step by step.

*s*

*A*

*t*

*B*

*C*

*D*

*E*

8

3

2

3

1

1

6

2

1

1

2

2

Answer: according to Dijkstra’s algorithm, we have:

S={s}, U={A,B,C,D,E,t}, find shortest edge with nodes from S to U:

**d(s, C)=2, through C:**

δ(C,A)=3, d(s, A)=5

δ(C,B)=1, d(s, B)=3

S={s, C}, U={A,B,D,E,t}, find shortest edge with nodes from S to U:

**d(s, B)=3, through B:**

δ(B,A)=1, d(s, A)=4

δ(B,t)=6, d(s, t)=9

S={s,C,B}, U={A,D,E,t}, find shortest edge with nodes from S to U:

**d(s, A)=4, through A:**

δ(A,D)=1, d(s, D)=5

δ(A,E)=1, d(s, E)=5

S={s,C,B,A}, U={D,E,t}, find shortest edge with nodes from S to U:

**d(s, D)=5, through D:**

δ(D, E)=2, 5+2=7>5

δ(D, t)=2, d(s, t)=7

S={s,C,B,A,D}, U={E,t}, find shortest edge with nodes from S to U:

**d(s, E)=5, through E:**

δ(E, s)=2, 5+2=7>0

S={s,C,B,A,D,E}, U={t}, find shortest edge with nodes from S to U:

**d(s, t)=7, through t:**

None

S={s,C,B,A,D,E,t}, U={}

So shortest paths from s to t: d(s, t)=7

**Q12.** Find a minimum spanning tree.

*A*

*B*

*C*

*D*

*E*

*F*

1

3

3

5

1

6

2

4

1

Answer:

Using Kruskal Algorithm:

M={}

Sorted edges weights:

w(A,B)=1, w(B,F)=1, w(D,E)=1, w(C,D)=2, w(A,E)=3, w(B,C)=3, w(C,F)=4, w(B,E)=5, w(B,D)=6

w(A,B)=1, no circle -> M={w(A,B)}, w(M)=1

w(B,F)=1, no circle -> M={w(A,B),w(B,F)}, w(M)=2

w(D,E)=1, no circle -> M={w(A,B),w(B,F),w(D,E)}, w(M)=3

w(C,D)=2, no circle -> M={w(A,B),w(B,F),w(D,E),w(C,D)}, w(M)=5

w(A,E)=3, no circle -> M={w(A,B),w(B,F),w(D,E),w(C,D),w(A,E)}, w(M)=8

w(B,C)=3, circle existed if add into M, next

w(C,F)=4, circle existed if add into M, next

w(B,E)=5, circle existed if add into M, next

w(B,D)=6, circle existed if add into M, end.

So the MST contains:

M={w(A,B),w(B,F),w(D,E),w(C,D),w(A,E)}, w(M)=8

The MST as bellow:

*A*

*B*

*C*

*D*

*E*

*F*

1

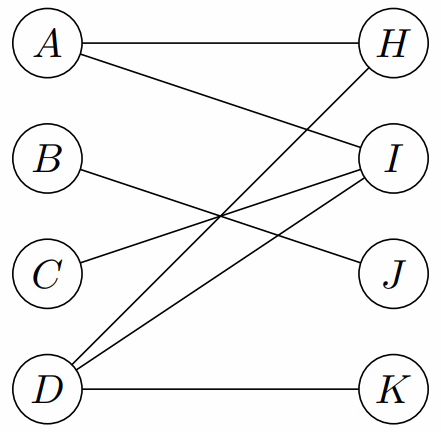
3

1

2

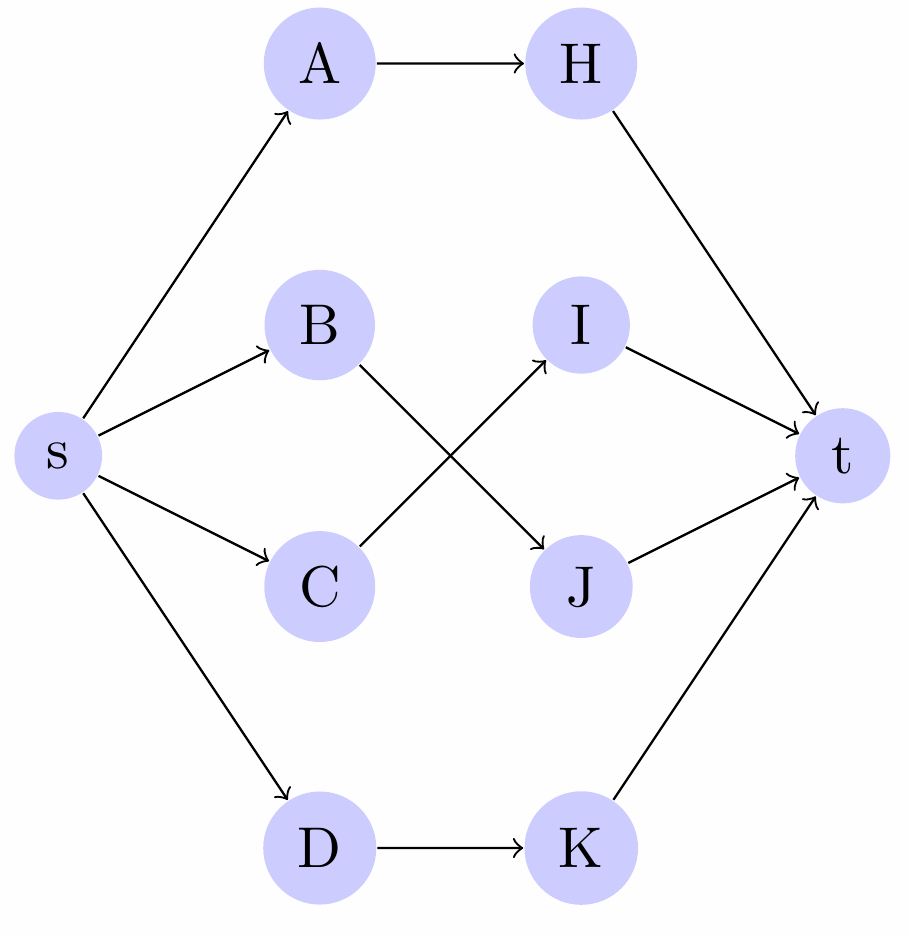
1

**Q13.** Find a maximum bipartite matching.



Answer:

We can use Ford-Fulkerson’s Algorithm to find Maximum Flow with a source s and a target t:



We can get a maximum bipartite matching from above flow:

