Dept. of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

ALGORITHMS
(EC31205)

THE SOLITARY ASSIGNMENT

AUTHOR: JAYA KISHNANI
ROLL NUMBER: 20EC30020
DATE: 10/11/2022

ANSWER -1 EXERCISE 0.1

a) $\mathrm{f(n) = n - 100}$ and $\mathrm{g(n) = n - 200}$

$$\frac{f(n)}{g(n)} = \frac{n-100}{n-200} \leq 101 \; for\; all\; n \neq 200$$

Therefore $f = O(g)$

Similarly,

$$\frac{g(n)}{f(n)} = \frac{\mathrm{n-200}}{\mathrm{n-100}} \leq 1 \; for\; all\; n \neq 200$$

Therefor $g = \Omega(\mathrm{f})$

Hence,

$$\mathbf{f = \Theta(g)}$$

b) $\mathrm{f(n) = n^{\frac{1}{2}}}$ and $\mathrm{g(n) = n^{\frac{2}{3}}}$

We know that $\frac{2}{3} > \frac{1}{2}$

Hence,

$$\mathbf{f = O(g)}$$

c) $\mathrm{f(n) = 100n + logn}$ and $\mathrm{g(n) = n + (logn)^2}$

We know that polynomial functions dominate over logarithmic function

Therefore, $f(n) = g(n) = O(n)$

Hence,

$$\mathbf{f = \Theta(g)}$$

d) $\mathrm{f(n) = nlogn}$ and $\mathrm{g(n) = 10nlog(10n)}$

$$g(n) = O(nlog10n) = O(nlog10 + nlogn) = \mathrm{O}(nlogn)$$

$$f(n) = \mathrm{O(nlogn + nlog10)} = O(nlog10n) = \mathrm{O(10nlog10n)}$$

Therefore, $f = O(g) and\; \mathrm{f} = \Omega(\mathrm{g})$

Hence,

$$\mathbf{f = \Theta(g)}$$

e) $\mathrm{f(n) = log2n}$ and $\mathrm{g(n) = log3n}$

$$f(n) = O(log3n) = O(log3 + logn) = O(log2 + \mathrm{logn}) = O(log2n)$$

$$g(n) = O(log2n) = \mathrm{O(log2 + logn)} = O(log3 + logn) = O(log3n)$$

Therefore, $f = O(g) and\; \mathrm{f} = \Omega(\mathrm{g})$

Hence,

$$\mathbf{f = \Theta(g)}$$

f) $f(n) = 10\log n$ and $g(n) = \log(n^2)$

$$g(n) = 2logn$$

We can observe that the $f(n)$ and $g(n)$ differ by constants

Hence,

$$\mathbf{f = \Theta(g)}$$

g) $f(n) = n^{1.01}$ and $g(n) = n\log^2 n$

$$f(n) = n.n^{0.01}$$

$$\frac{f(n)}{g(n)} = \frac{n^{0.01}}{\log^2 n}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{0.01}}{\log^2 n}$$

Using L-Hospital's rule

$$\lim_{n\to\infty} \frac{0.01n^{-0.99}}{2\left(\frac{logn}{n}\right)} = \lim_{n\to\infty} \left(\frac{0.01n^{0.01}}{2\log n}\right) = \lim_{n\to\infty} \left(\frac{0.01^2 n^{0.01}}{2}\right) = \infty$$

Hence,

$$\mathbf{f = \Omega(g)}$$

h) $f(n) = \frac{n^2}{\log n}$ and $g(n) = n(\log n)^2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{n^2}{logn}}{n(logn)^2} = \frac{n}{(\log n)^3}$$

Using L-Hospital's rule,

$$\lim_{n\to\infty} \frac{n}{3(logn)^2} = \frac{n}{6logn} = \frac{n}{6} = \infty$$

Hence,

$$\mathbf{f = \Omega(g)}$$

i) $f(n) = n^{0.1}$ and $g(n) = (\log n)^{10}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{0.1}}{(logn)^{10}}$$

Using L-Hospital's rule,

$$\lim_{n\to\infty} \frac{n^{0.1}}{(logn)^{10}} = \lim_{n\to\infty} \frac{0.1n^{-0.99}.n}{10(logn)^9} = \frac{0.1n^{0.1}}{10(logn)^9} = \cdots = \lim_{n\to\infty} \frac{0.1^9 n^{0.1}}{10!\,logn} = \infty$$

Hence,

$$\mathbf{f = \Omega(g)}$$

j) $\mathrm{f(n) = logn^{logn}}$ and $\mathrm{g(n)} = \frac{\mathrm{n}}{\mathrm{logn}}$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{(\mathrm{logn})^{\mathrm{logn}}}{\frac{\mathrm{n}}{\mathrm{logn}}} = \frac{(\mathrm{logn})^{\mathrm{logn+1}}}{n}$$

Using L-Hospital's rule,

$$\lim_{n\to\infty}\frac{(\mathrm{logn}+1)\mathrm{logn^{logn}}}{1} = \infty$$

Hence,

$$\mathbf{f = \Omega(g)}$$

k) $\mathrm{f(n)} = \sqrt{\mathrm{n}}$ and $\mathrm{g(n) = (logn)^3}$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{\sqrt{\mathrm{n}}}{(logn)^3}$$

Using L-Hospital's rule,

$$\lim_{n\to\infty}\frac{\sqrt{\mathrm{n}}}{6(logn)^2} = \lim_{n\to\infty}\frac{\sqrt{\mathrm{n}}}{24\mathrm{logn}} = \lim_{n\to\infty}\frac{\sqrt{\mathrm{n}}}{48} = \infty$$

Hence,

$$\mathbf{f = \Omega(g)}$$

l) $\mathrm{f(n) = n^{\frac{1}{2}}}$ and $\mathrm{g(n) = 5^{\log_2 n}}$

$$\mathrm{g(n) = n^{\log_2 5}}$$

Since, $\log_2 5\ (2.322) > \frac{1}{2}$

Hence,

$$\mathbf{f = O(g)}$$

m) $\mathrm{f(n) = n2^n}$ and $\mathrm{g(n) = 3^n}$

$$g(n) = (1+2)^n = C_1^n 2^n + C_2^n 2^{n-1} + C_3^n 2^{n-2} + \cdots$$

Neglecting higher order terms,

$$g(n) = \Theta(2^n) \text{ and } \mathrm{f(n) = n2^n}$$

Hence,

$$\mathbf{f = O(g)}$$

n) $\mathrm{f(n) = 2^n}$ and $\mathrm{g(n) = 2^{n+1}}$

$$g(n) = 2.2^n$$

We neglect coefficient, hence

$$\mathbf{f = \theta(g)}$$

o) $\mathrm{f(n) = n!}$ and $\mathrm{g(n) = 2^n}$

For n>2

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{\mathrm{n!}}{2^{\mathrm{n}}} = \lim_{n\to\infty}\left(\frac{n}{2}\right)\left(\frac{n-1}{2}\right)\left(\frac{\mathrm{n}-2}{2}\right)\dots\frac{3}{2}.\frac{2}{2}.\frac{1}{2} = \infty$$

Hence,

$$\mathbf{f = \Omega(g)}$$

p) $\mathrm{f(n) = (logn)^{logn}}$ and $\mathrm{g(n) = 2^{(\log_2 n)^2}}$

$$g(n) = (2^{\log_2 n})^{\log_2 n} = \mathrm{n}^{\log_2 n} = \frac{1}{n^{log2^{logn}}} = (\mathrm{n}^{3.322})^{logn}$$

We know, $O(n) > O(logn) => \mathrm{O}(\mathrm{n}^{3.322}) > O(logn)$

So, $(n^{3.322})^{logn} > O(logn^{logn})$

Hence,

$$\mathbf{f = O(g)}$$

q) $\mathrm{f(n)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{i^k}$ and $\mathrm{g(n) = n^{k+1}}$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{1^{\mathrm{k}} + 2^k + 3^k + \dots + \mathrm{n^k}}{\mathrm{n^{k+1}}} = \lim_{n\to\infty}\left(\frac{1^k}{n^{k+1}}\right)\left(\frac{2^k}{n^{k+1}}\right)\left(\frac{3^k}{n^{k+1}}\right)\dots\left(\frac{\mathrm{n}^k}{n^{k+1}}\right) = \infty$$

Hence,

$$\mathbf{f = O(g)}$$

ANSWER-2 EXERCISE 1.1

Let x, y and z be the single digit numbers in base b.

Then we can say that x, y, z can be atmost (b-1). Since, for k digits in base b we can express numbers up to $(\mathrm{b^k} - 1)$.

Let the sum of three single digits be S

So

$\mathrm{S = x + y + z}$

$\mathrm{S = (b-1) + (b-1) + (b-1) = 3 * b - 3 = 3 * (b-1)}$

So for $\mathrm{b = 2, S = 3 * (2-1) = 3 = 11}$, which is atmost 2 digit long.

For $\mathrm{b \geq 3, S = 3 * (b-1) \leq b * (b-1) < b\hat{}2}$

So, now $\mathrm{S < b^2}$

So, we can say that $\mathrm{S < b^2}$ can be represented in 2 digits.

Hence we can say that sum of three signal digit number is at most two digit long.

ANSWER-2 EXERCISE 2.3

a) Given : $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$

As described in the question, let us assume $O(n) \le cn$ for some constant c.

The above expression can be written as:

$$T(n) \le 3T\left(\frac{n}{2}\right) + cn$$

$$T(n) \le 3\left[3T\left(\frac{n}{4}\right) + \frac{cn}{2}\right] + cn$$

$$T(n) \le 9T\left(\frac{n}{4}\right) + \frac{3cn}{2} + cn$$

$$T(n) \le 9\left[3T\left(\frac{n}{8}\right) + \frac{cn}{4}\right] + \frac{3cn}{2} + cn$$

$$T(n) \le 27T\left(\frac{n}{8}\right) + \frac{9cn}{4} + \frac{3cn}{2} + cn$$

And the series continues

So we can observe the pattern below,

$$T(n) \le 3^k T\left(\frac{n}{2^k}\right) + \left(\left(\frac{3}{2}\right)^k - 1\right)2cn$$

Putting , $\frac{n}{2^k} = 1$

We get, $k = \log_2 n$

Putting $k = \log_2 n$, we get

$$T(n) \le 3^{\log_2 n} T(1) + \left(\frac{3^{\log_2 n}}{2^{\log_2 n}} - 1\right)2cn$$

$$T(n) \le n^{\log_2 3} T(1) + \left(n^{\log_2 3} - n\right)2c$$

As $T(1) = O(1) = k'$ (some constant),

Hence

$$\mathbf{T(n) \le O(n^{\log_2 3})}$$

b) Given: $T(n) = T(n-1) + O(1)$

It can be recursively written as,

$$T(n) = T(n-1) + O(1)$$

$$T(n) = T(n-2) + 2O(1)$$

$$T(n) = T(n-3) + 3O(1)$$

And so on

So we can observe the pattern as,

$$T(n) = T(n-k) + kO(1)$$

Let's say $n - k = 1$

We get $k = n - 1$

Putting $k = n - 1$, we get

$$T(n) \le T(1) + (n-1)O(1)$$

As $T(1) = O(1) = k'$ (some constant), therefore

$$T(n) \le c(n-1) + c$$

$$\mathbf{T(n) \le O(n)}$$

ANSWER-2 EXERCISE 2.5

b) $T(n) = 5T\left(\frac{n}{4}\right) + n$

Using masters theorem,

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d$$

In this case a=5, b=4, c=1, d=1

Since, we know $\frac{a}{b^d} = \frac{5}{4} > 1$

$$T(n) = c\frac{\left(an^{\log_b a} - b^d n^d\right)}{a - b^d} = 5n^{\log_4 5} - 4n$$

Hence,

$$\mathbf{T = \Theta(5n^{1.161} - 4n)}$$

e) $T(n) = 8T\left(\frac{n}{2}\right) + n^3$

Using masters theorem,

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d$$

In this case a=8, b=2, c=1, d=3

Since, we know $\frac{a}{b^d} = \frac{8}{8} = 1$

$$T(n) = cn^d(\log_b n + 1) = n^3(\log_2 n + 1)$$

Hence,

$$\mathbf{T(n) = \Theta(n^3(\log_2 n + 1))}$$

h) $T(n) = T(n-1) + n^c$, where $c \ge 1$ is a constant

Hence, $T(n) = n^c + (n-1)^c + (n-2)^c + \cdots + T(n-k)$

$$T(n) = T(n-k) + \sum_{k=1}^{k} (n - (i-1))^c$$

Let n-k =0 ➔ k=n

Substituting k = n, we get

$$T(n) = T(0) + \sum_{k=1}^{\mathrm{k}} (n - (\mathrm{i} - 1))^c$$

As, $T(1) = O(1)$ = k' (some constant)

$$T(n) = k' + \sum_{k=1}^{\mathrm{k}} (n - (\mathrm{i} - 1))^c$$

$$T(n) \leq n^{c+1}$$

$$\mathbf{T(n) = O\left(n^{c+1}\right)}$$

i) $T(\mathrm{n}) = \mathrm{T(n-1) + c^n}$, where $\mathrm{c} \geq 1$ is a constant

Hence, $\mathrm{T(n) = c^n + c^{n-1} + \cdots + c^2 + k}$, where $\mathrm{k = T(1)}$

$$\mathrm{T(n) = k + \sum_{i=2}^{n} c^t = k + \frac{c^2(c^{n-1} - 1)}{c - 1}}$$

Ignoring the constant terms, we get

$$\mathbf{T(n) = \Theta(c^{n-1} - 1)}$$

k) $\mathrm{T(n) = T\left(\sqrt{n}\right) + 1}$

Let $n = 2^k$,

$$k = \log_2 n$$

$$T\left(2^k\right) = T\left(2^{\frac{k}{2}}\right) + 1$$

Let $T(n) = T\left(2^k\right) = T'(k)$

Then,

$$T'(k) = T'\left(\frac{\mathrm{k}}{2}\right) + 1$$

Using master's theorem,

$$T'(n) = aT'\left(\frac{k}{b}\right) + ck^d$$

Where a=1, b=2, c=1, d=0

Since $\frac{a}{b^d} = \frac{1}{2^0} = 1$

$$T'(k) = ck^d(\log_b k + 1) = \log_2 k + 1$$

Let us replace k and T'(k) as assumed above,

$$T(n) = \log_2 \log_2 n + 1$$

Hence,

$$\mathbf{T(n) = \Theta(\log_2 \log_2 n + 1)}$$

ANSWER-5 EXERCISE 2.7

We know that for any polynomial of nth degree

$$\text{Sum of roots} = -\frac{\text{coefficients of } (n-1)\text{th degree term}}{\text{cofficient of nth degree term}}$$

$$\text{Product of roots} = (-1)^n \frac{\text{constant term}}{\text{coefficient of nth degree term}}$$

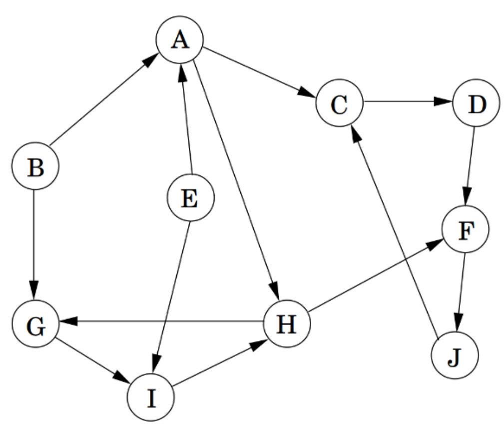
The given polynomial is : $x^n - 1 = 0$

Therefore the coefficient of (n-1)th term is zero and hence **sum of nth roots of unity is zero**.

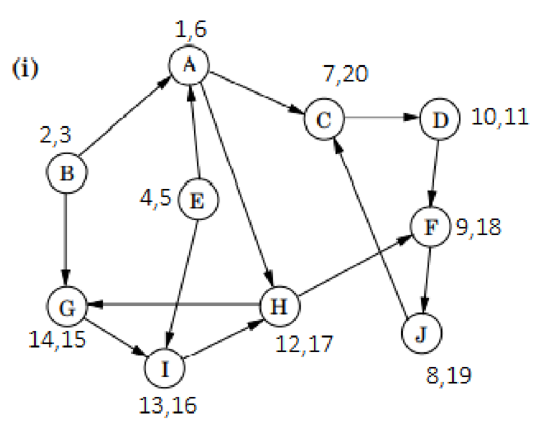
The constant term is -1 and coefficient of nth degree term is 1.

Therefore, **if n is odd the product of roots is 1, and if n is even the product of roots is -1**.

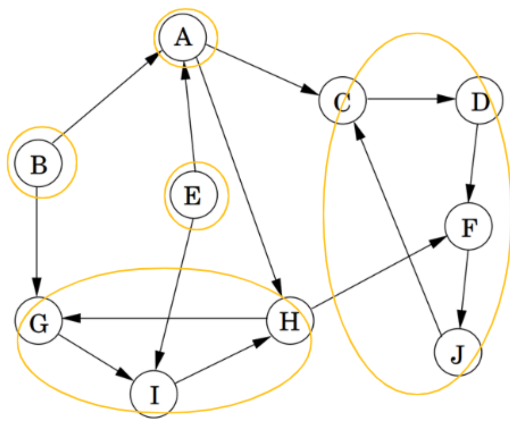
ANSWER-6 EXERCISE 3.4



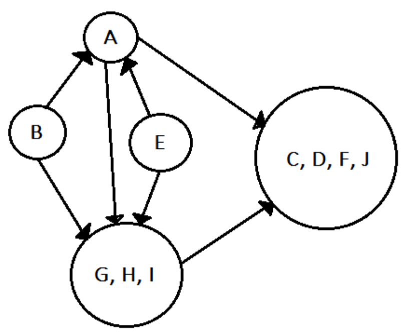
If we run DFS on G^R, we get following pre and post visit numbers



Now if we run DFS on G in decreasing order of post no's we get, the following SCC's



a) The SCC's are determined in order.
   {C, D, F, J}, {H, G, I}, {A}, {E}, {B}
b) The source SCC's are {E}, {B} and the sink SCC is {C, D, F, J}.
c) Meta graph is shown in the following figure as:



d) To make it strongly connected one directed edge from E to B and another from C to E should be added. So minimum two edges must be added.

ANSWER-7 EXERCISE 3.6

a) Proof by principle of induction:

   Let us assume that let number of edges in a directed graph G(V,E) be n.
   Let us assume that $X(n) = 2n$ where $X(n) = \sum_{v \in V} \deg(v)$.

   When there is no edge, then n=0. All the nodes in G have zero degree.

Therefore $X(0) = 2 * 0 = 0$

Let us assume that $X(n)$ is true, then we have to show that $X(n+1)$ is also true, i.e.

$$2(n+1) = \sum_{v \in V} \deg(v)$$

In graph G with n+1 edges, if we randomly remove an edge, we obtain a subgraph G'(V', E') for which we can assume $X(n) = 2n = \sum_{v \in V'} \deg(v)$. The vertex of G and G' are same, therefore V = V'

$$\sum_{v \in V'} \deg(v) = \sum_{v \in V} \deg(v) = 2n$$

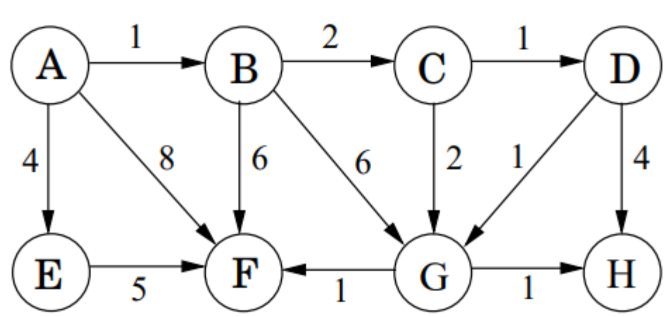
If we add edge to G', we obtain G. Since every edge connects two vertices it adds 2 to the total number of degrees.
Hence,

$$\sum_{v \in V'} \deg(v) = \sum_{v \in V} \deg(v) = 2n + 2 = 2(n+1) = X(n+1)$$

This proves that X(n+1) is true. Therefore inductively we can say that 2|E| = $\sum_{v \in V} \deg(v)$ for a undirected graph G(E,V).

b) From the proof of the above part a, we can deduce that sum of degrees for undirected graph is even.
Let us assume that the number of vertices having odd degree is odd, therefore the sum of degrees of all vertices having odd degree is odd.
Then we obtain sum of degrees of all vertices having even degree should be odd (even subtracted with odd results in odd), which is impossible.
Therefore, from the above argument we can say that for an undirected graph, there must be an even number of vertices whose degree is odd.

ANSWER-8 EXERCISE 4.1

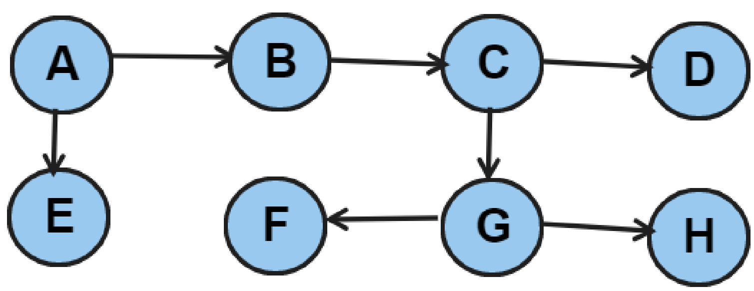


a) Since the starting vertex is A.
   Let the initial cost at A be 0 and the cost of A to E be 4, A to F be 8, A to B be 1.
   Let the cost of the other remaining vertices be ∞.

A	B	C	D	E	F	G	H
0	∞	∞	∞	∞	∞	∞	∞
0	1	∞	∞	4	8	∞	∞
0	1	3	∞	4	7	7	∞
0	1	3	4	4	7	5	∞
0	1	3	4	4	7	5	8
0	1	3	4	4	7	5	8
0	1	3	4	4	6	5	6
0	1	3	4	4	6	5	6

   The node coloured in yellow after each iteration is taken as the node visited.

b) Shortest path tree:



ANSWER-9 EXERCISE-4.14

Given input: Strongly connected directed graph G = (V,E) with positive edge lengths and node $v_o \in V$.
Output: To find shortest path between all pairs of nodes passing through v_o.

Algorithm:

1) Apply Dijkstra's shortest path algorithm on G with v_o as start node. This provides us with shortest path from v_o to all other nodes in G.

2) Apply Dijkstra's shortest path algorithm on G^R with v_o as start node. This provides us with shortest path from v_o to all other nodes in G^R. So on reversing back the paths, we determine shortest path from all other nodes towards v_o in G.
3) We iterate over all pairs to obtain shortest path between all pairs of nodes. For example, from node $v_1$to v_2, we first find shortest path from node v_1 to v_o using step 2 and then from v_o to v_2 using step1. Then we add both the paths to get shortest path from v_1 to v_2 passing through v_o.

ANSWER-10 EXERCISE-5.1

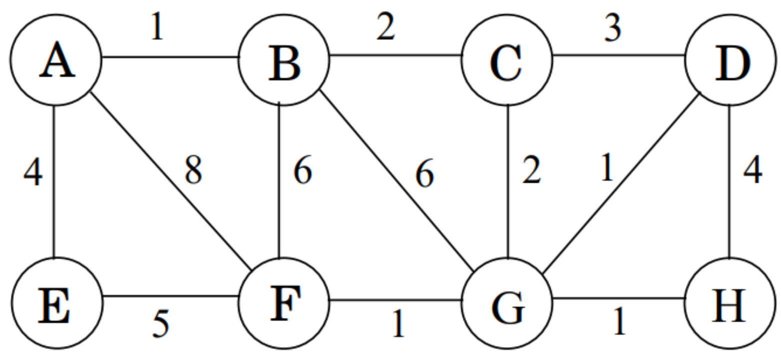
Firstly, We sort the weights of the graph in increasing order.

Weight	Source	Destination
1	A	E
1	E	F
2	E	B
2	B	F
3	F	G
3	G	H
4	C	G
5	B	C
5	F	C
5	G	D
6	A	B
6	C	D
7	D	H

By Kruskal's algorithm, MST is
A-E, E-F, E-B, F-G, G-H, G-C, G-D

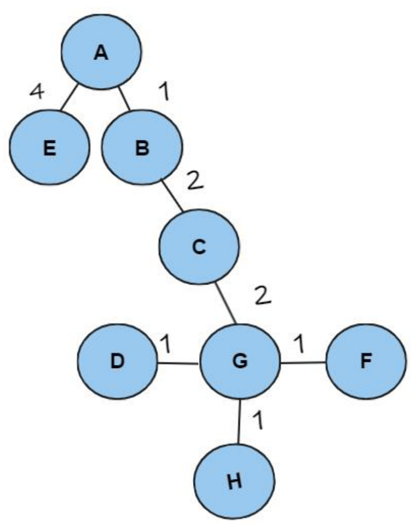
a) The cost of the MST = 1+1+2+3+3+4+5 = **19**
b) The minimum spanning trees the graph have will be **2**. Since initially we can choose between edge A-E and E-F.
c) If the edges are sorted by weight in the order
A-E, E-F, E-B, B-F, F-G, G-H, C-G, F-C, G-D, B-C, A-B, C-D, D-H
Then we add the edges in order given below,
1) A-E
2) E-F
3) E-B
4) F-G
5) G-H
6) G-C
7) G-D

ANSWER-11 EXERCISE 5.2



Input: A connected undirected graph $G = (V, E)$ with edge weights w_e

Set S	A	B	C	D	E	F	G	H
{}	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A		1/A	∞/nil	∞/nil	4/A	8/A	∞/nil	∞/nil
A,B			2/B	∞/nil	4/A	6/B	6/B	∞/nil
A,B,C				3/C	4/A	6/B	2/C	∞/nil
A,B,C,G				1/G	4/A	1/G		1/G
A,B,C,G,D					4/A	1/G		1/G
A,B,C,G,D,F					4/A			1/G
A,B,C,G,D,F,H					4/A			
A,B,C,G,D,F,H,E								



From the above table the cost of the MST obtained is 12.

ANSWER-12 EXERCISE-5.6

If we consider cut-property, then for a subset S $(S \in V)$, the edge with minimum cost that connects S and V-S is a part of MST. If there is more than one MST, then for some subset S and V-S there exist two or more number of edges having equal and smallest cost, which will be the part of different MST's. But for the given graph G, each edge is unique. This implies that for any subset $S \in V \; or \; S \in V - S$, there will be a unique edge with minimum cost and all the unique edges will be the part of the MST.

Hence proved that all the edge weights of an undirected graph should be distinct for a unique MST.

ANSWER-13 EXERCISE -6.2

For minimizing the penalty, we need to minimize the penalties of sub-problems.

Let $\min P(j)$ be considered as minimum penalty gained upto hotel a_j where $0 \leq j \leq n$ as a subproblem. Then the minimum penalty at final hotel a_n be given as $\min P(n)$.

Let's assume that the trip stopped at location a_j and a_i is the previous stop where $0 \leq \mathrm{i} < \mathrm{j}$, then

$$\mathbf{min\,P(j) = min\{P(i) + \left(200 - \left(a_j - a_i\right)\right)^2\}}$$

Pseudo code:

```
min P(0) = 0
for i = 1, 2, .... n:
    min P(i) = inf
    i = i+1
for j = 1, 2, ... n:
    for i = 0, 1, ... , j-1:
        min P(j) = min{ min P(j), min{P(i) + (200-(a[j]-a[i]))**2}}
        i = i+1
    j = j+1
return min P(n)
```

Since there are n subproblems and each sub problem takes $O(n)$ time to solve. Therefore, time complexity of the algorithm is $O(n^2)$.

ANSWER-14 EXERCISE-6.26

Sequence alignment

Here the score matrix δ of size [|∑| +1] x [|∑| +1] (where ∑ denotes the alphabets in the string) determines closeness in strings. We can say that the two strings can have highest scoring alignment if the substrings have highest score alignment.

Input: Two strings x[1….m] and y[1…..n] and scoring matrix δ of size [|∑| +1] x [|∑| +1]
Output: Highest scoring alignment of the given strings x and y

Here, we would be looking at the edit distance between x[1:i] and y[1:j] and let us consider this sub-problem as f(i,j), So our final objective will be to compute f(m,n)

So there are three possible cases to align strings x and y:

1) x[i] aligns to y[j], which incurs cost of $\mathrm{\delta}(\mathrm{x}[\mathrm{i}], \mathrm{y}[\mathrm{j}])$ and we need to calculate $f(i-1, j-1)$
2) x[i] aligns to a gap, which incurs cost of $\mathrm{\delta}(\mathrm{x}[\mathrm{i}], -)$ and we need to calculate $f(i-1, j)$
3) y[i] aligns to a gap, which incurs cost of $\mathrm{\delta}(-, \mathrm{y}[\mathrm{j}])$ and we need to calculate $f(i, j-1)$

We assume that $f(i, j-1), f(i-1, j)$ and $\mathrm{f}(\mathrm{i}-1, \mathrm{j}-1)$ are optimal scores of the sub-problem

Then $\mathbf{f}(\mathbf{i}, \mathbf{j}) = \mathbf{max}\{\mathbf{f}(\mathbf{i}-\mathbf{1}, \mathbf{j}-\mathbf{1}) + \boldsymbol{\delta}(\mathbf{x}[\mathbf{i}], \mathbf{y}[\mathbf{j}]) ,\ \mathbf{f}(\mathbf{i}-\mathbf{1}, \mathbf{j}) + \boldsymbol{\delta}(\mathbf{x}[\mathbf{i}], -) ,\ \mathbf{f}(\mathbf{i}, \mathbf{j}-\mathbf{1}) + \boldsymbol{\delta}(-, \mathbf{y}[\mathbf{j}])\ \}$

For the smallest sub-problems, the base cases, we have $f(0, j)\ and\ f(i, 0)$, this is edit-distance between zero-length prefix of x with first j alphabets of y and vice-versa respectively.

Pseudo-code:

```
f(0,0) = 0
for i = 1, 2 ... m:
    f(i,0) = δ(x[i],−) + δ(x[i − 1],−)
for j = 1, 2, ... n:
    f(0,j) = δ(− ,y[j]) + δ(−,y[j − 1])
for i = 1, 2, ...m:
    for j = 1, 2, ... n:
        f(i,j) = max( δ(x[i],y[i]) + f(i-1, j-1), δ(− ,y[i]) + f(i, j-1), δ(x[i],−)
f(i-1,j))
return f(m,n)
```

The above algorithm has time and space complexity as $\mathbf{O}(\mathbf{mn})$. Since we fill the 2-D matrix row by row and each entry takes O(1) time to fill up.

ANSWER-12 EXERCISE-7.5

a) The problem can be modelled as linear program in two variables.
Let x1 be the number of packages of Frisky Pup
Let x2 be the number of packages of Husky hound

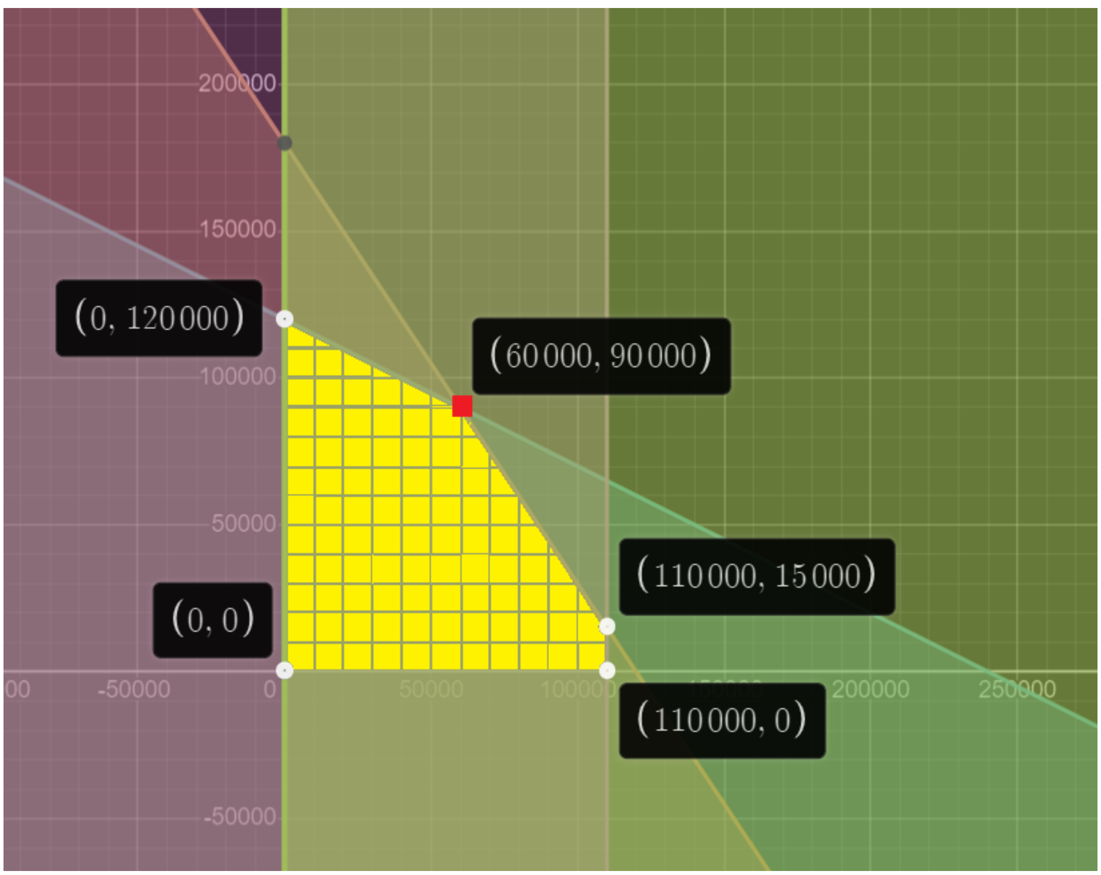
Objective function:

$$\max 7\text{x1} + 6\text{x2} - \text{x1} - 2 * 1.5\text{x1} - 2\text{x2} - 2\text{x2} - 1.4\text{x1} - 0.6\text{x2} = 1.6\text{x1} + 1.4\text{x2}$$

Constraints:

$$\text{x1} + 2\text{x2} \leq 240{,}000$$
$$1.5\text{x1} + \text{x2} \leq 180{,}000$$
$$\text{x1} \leq 110{,}000$$
$$\text{x1}, \text{x2} \geq 0$$

b) The graph of the above set of constraints maximizing the profit can be plotted as shown below



The yellow region in the graph indicates the feasible region and the optimal point is (60000, 90000) which maximizes the profit.

Hence, the maximum profit obtained is 1.6*(60000) + 1.4*(90000) = **$222,000**.