D.Naveen Kumar
18A51A0518
CSE - A
DAA

Assignment - 1

1) Calculate the time complexity for the following expressions by using Big-oh notation.

1) $4n+3$
2) $100n+9$
3) $10n^2+7n+2$
4) $1000n^2+100n-4$
5) $8*2^n+n^2$

Big-oh notation:- It is stated as $f(n)=O(g(n))$ if and only if there exists positive constants c and n_0 such that $f(n) \le c*g(n)$ for all $n, n \ge n_0$.

1) $4n+3$

$4n+3 \le 6n \ \forall n \ge 2$

so, "$4n+3$" time complexity is $O(n)$.

2) $100n+9$

$100n+9 \le 109n \ \forall n \ge 1$

so, time complexity for $100n+9$ is $O(n)$.

3) $10n^2+7n+2$

$10n^2+7n+2 \le 20n^2 \ \forall n \ge 1$

so, time complexity for $10n^2+7n+2$ is $O(n^2)$

4) $1000n^2+100n-4$

$1000n^2+100n-4 \le 2000n^2 \ \forall n \ge 1$

so, time complexity is $O(n^2)$

5) $8*2^n+n^2$

$8*2^n+n^2 \le 19*2^n \ \forall n \ge 1$

so, time complexity is $O(2^n)$

2) State the various pseudo-code conventions for algorithm specification.

Pseudo-code Conventions:-

1) An identifier must always starts with a letter.
   Ex: priya12 (Valid
   1priya (invalid)

2) A set of statements (Blocks) must be included within braces ({ and })

3) Comment lines must be represented with "//".

4) Assignment is done by using the operator ":=".

5) Elements of multi-dimensional arrays are accessed using square braces ("[" and "]").

6) Boolean values (True and False), logical operators (and, or, not) Relational operators (greater than, less than ... etc) are used.

7) looping statements like: for, while and repeat-until are used.

Syntax * for:

```
for variable := value 1 to value 2 step step do
  {
    <st-1>
      :
    <st-n>
  }
```

Syntax of repeat-until:

```
repeat
    <st-1>
      :
    <st-n>
until <condition>
```

8) Individual data items of a record can be accessed with "→" and period.

9) conditional statements like : if, case etc. are used

<u>Syntax of if:</u>

If <condition>
then
<statement>

<u>Syntax of case:</u>

case
{
:< condition 1>
<st - 1>
:
:
:< condition n>
<st - n>
:else : <st - n+1>
}

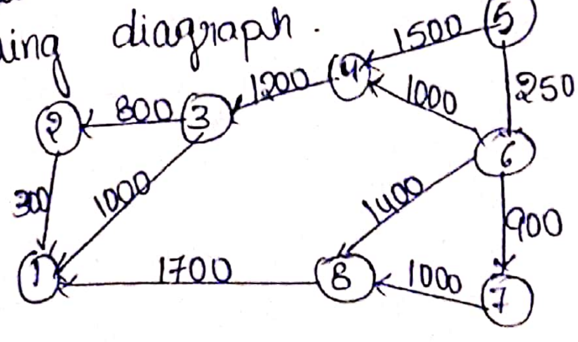
<u>Syntax of if - else</u> :

if <condition>
then
<st - 1>
else
<st - 2>

10) Both input and output operations are done using the instructions read and write respectively.

11) There is only one type of procedure called " Algorithm". An algorithm consists of heading and body.

③ Calculate the shortest path and the corresponding distance from the source node (5) to the destination nodes (1,2,3,4,6,7,8) for the following diagraph.



& <u>procedure</u>: In this single source shortest path problem we construct a table in which we represent it with "∞" if there exists no path b/t two nodes. if the path exists then we go on comparing the current cost with it previous cost value if it is less then we have to replace it with

→ we have to go on repeat this process until we are left with "n-1" elements.

Matrix:

	1	2	3	4	5	6	7	8
1	0							
2	300	0						
3	1000	800	0					
4			1200	0				
5				1500	0	250		
6				1000		0	900	1400
7								
8	1700						0	1000
								0

Table:

Set / Value	1	2	3	4	6	7	8
	∞ (5-1)	∞ (5-2)	∞ (5-3)	1500 (5-4)	250 (5-6)	∞ (5-7)	∞ (5-8)
{5}	∞ (5-6-1)	∞ (5-6-2)	∞ (5-6-3)	1250 (5-6-4)	250 (5-6)	1150 (5-6-7)	1650 (5-6-8)
{5,6}	∞ (5-6-7-1)	∞ (5-6-7-2)	∞ (5-6-7-3)	~~∞ (5-6-7-4)~~ 1250 (5-6-4)	~~∞ (5-6-7-6)~~ 250 (5-6)	1150 (5-6-7-7)	~~2150 (5-6-7-8)~~ 1650 (5-6-8)
{5,6,7}	∞ (5-6-4-1)	∞ (5-6-4-2)	2450 (5-6-4-3)	1250 (5-6-4)	~~∞ (5-6-4-6)~~ 250 (5-6)	~~∞ (5-6-7-7)~~ 1150 (5-6-7)	~~∞ (5-6-4-8)~~ 1650 (5-6-8)
{5,6,7,4}	3350 (5-6-8-1)	∞ (5-6-8-2)	~~∞ (5-6-8-3)~~ 2450 (5-6-4-3)	~~∞ (5-6-8-4)~~ 1250 (5-6-4)	~~∞ (5-6-8-6)~~ 250 (5-6)	~~∞ (5-6-8-7)~~ 1150 (5-6-7)	1650 (5-6-8)
{5,6,7,4,8}	~~3450 (5-6-4-3-1)~~ 3350 (5-6-8-1)	3250 (5-6-4-3-2)	2450 (5-6-4-3-3)	~~∞ (5-6-4-3-4)~~ 1250 (5-6-4)	~~∞ (5-6-4-3-6)~~ 250 (5-6)	~~∞ (5-6-4-3-7)~~ 1150 (5-6-4-3-7)	~~∞ (5-6-4-3-8)~~ 1650 (5-6-8)

{5,6,7,4,8,3}	3350 (5-6-4-3-2-1)	3250 (5-6-4-3-2)	∞ (5-6-4-3-2-3) 2450 (5-6-4-3)	∞ (5-6-4-3-2-4) 1250 (5-6-4)	∞ (5-6-4-3-2-6) 250 (5-6)	∞ (5-6-4-3-2-7) 1150 (5-6-4-3-7)	∞ (5-6-4-3-2-8) 1650 (5-6-8)
{5,6,7,4 8,3,2}	3350 (5-6-8-1-1)	∞ 3250 (5-6-4-3-2)	∞ 2450 (5-6-4-3)	∞ (5-6-4)	∞ (5-6)	∞ (5-6-4-7)	∞ (5-6-8)

Final shortest path = 5-6-7-4-8-3-2

corresponding distance from the source node

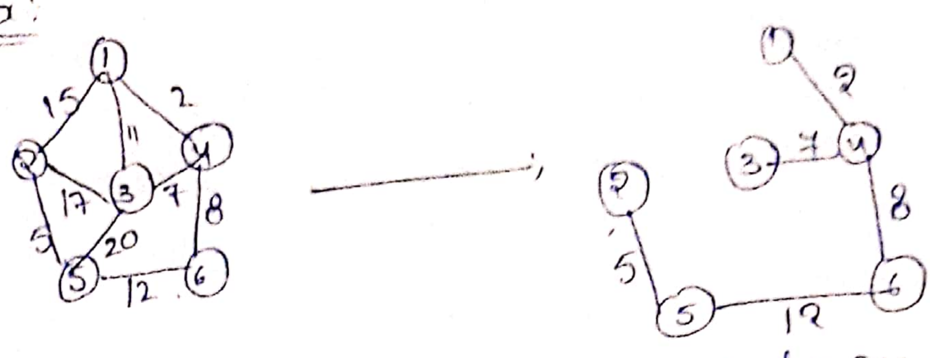
(5) = 5-6-7-4-8-3-2

(4) Develop the Kruskal algorithm for minimum cost spanning tree.

A Kruskal's Algorithm:

```
Algorithm Kruskal(E, cost, n, t)
{
contruct a min heap out of the edge costs using heapify;
for i:=1 to n do parent[i]:=-1;
i:=0; mincost:=0.0;
while ((i<n-1) and (heap not empty)) do
{
  delete a minimum cost edge (u,v) from the heap and
  reheapify using adjust;
  j:=find(u); k:=find(v);
  if (j≠k) then
  {
   i:=i+1;
   t[i,1]:=u, t[1,2]:=v;
   mincost:=mincost+cost[u,v];
   union(j,k);
  }
}
if (i≠n-1) then write("No spanning tree");
else return mincost;
}
```

Ex:



cost = 34

(3) Solve the following knapsack problem by using dynamic programming

$(P_1, P_2, P_3, P_4, P_5) = (20, 5, 10, 7, 15)$

$(w_1, w_2, w_3, w_4, w_5) = (2, 3, 1, 2, 3)$

capacity of knapsack = 5

0/1 knapsack problem:-

* For the knapsack problem using dynamic method, we consider either "0" or "1", but fractions are not considered.

i	P_i	W_i
1	20	2
2	5	3
3	10	1
4	7	2
5	15	3

For this problem
optimal solution = 1 0 1 1 0
profit = 37.

Using dynamic programming:-

$S^0 = \{0, 0\}$

S^0_1 = add 1st tuple to S^0

$= \{(20,2) + \{0,0\}\}$

$S^0_1 = \{(20,2)\}$

S^1 = merge S^0 and S^0_1

$= \{(0,0), (20,2)\}$

$S^1_1 = \{(5,3) + S^1\}$

$S^1_1 = \{(5,3), (25,5)\}$

$S^2 = \{(0,0),(20,2),(5,3),(25,5)\}$

$S_1^2 = \{(10,1) + S^2\}$

$S_1^2 = \{(10,1),(30,3),(15,4),(35,6)\}$

S^3 = merge S^2 and S_1^2.

$S^3 = \{(0,0),(20,2),(5,3),(25,5),(10,1),(30,3),(15,4),(35,6)\}$

$S_1^3 = \{(7,2) + S^3\}$

$S_1^3 = \{(7,2),(27,4),(12,5),(32,7),(17,3),(37,5),(22,6),(42,8)\}$.

$S^4 = \{(0,0),(20,2),(5,3),(25,5),(10,1),(30,3),(15,4),(35,6),(7,2),(27,4),$
$(12,5),(32,7),(17,3),(37,5),(22,6),(42,8)\}$

$S_1^4 = \{(15,3) + S^4\}$

$S_1^4 = \{(15,3),(35,5),(20,6),(40,8),(25,4),(45,6),(30,7),(50,9),(22,5),$
$(42,7),(27,8),(47,10),(32,6),(52,8),(37,9),(57,11)\}$.

$S^5 = \{(0,0),(20,2),(5,3),(25,5),(10,1),(30,3),(15,4),(35,6),(7,2),(27,4),(12,5),$
$(32,7),(17,3),(37,5),(22,6),(42,8),(15,3),(35,5),(20,6),(40,8),(25,4)$
$(45,6),(30,7),(50,9),(22,5),(42,7),(27,8),(47,10),(32,6),(52,8),$
$(37,9),(57,11)\}$.

Now, max$(P,W) = (37,5)$ occurred in $\underline{\underline{S^3}}$

so, $(37,5)$ – (3^{rd} tuple)

$= (37,5) - (10,1) = (27,4)$

Here, $(27,4)$ appeared in 4^{th} tuple so;

$(27,4)$ – 4^{th} tuple

$(27,4) - (7,2) = (20,2)$

Here, $(20,2)$ appeared in $\underline{S^1}$ so,

$(20,2) - (20,2) = \{0,0\}$.

Hence, optimal solution = [1 0 1 1 0]