Internal - 02

DAA (18CS42)

Shahul Hameed.S
1KN18CS097
CSE 'A' Sec

1a.

```
Quick sort (low, high)
{
 if low < high then
  {
    j := partition(a, low, high+1);
    //j is the position
    quicksort (low, j-1);
     quicksort (j+1, high);
  }
}
```

Best Case:- In the best case, the pivot is in the middle.

$$T(n) = 2T(n/2) + Cn \quad \text{---} \quad (1)$$

Divide both sides by n

$$\frac{T(n)}{n} = \frac{T(n/2)}{n/2} + C \quad \text{---} \quad (2)$$

Substitute $n = n/2$ in above eqn

$$\frac{T(n/2)}{n/2} = \frac{T(n/4)}{n/4} + C \quad \text{---} \quad (3)$$

Substitute $n = n/4$ in above eqn

$$\frac{T(n/4)}{n/4} = \frac{T(n/8)}{n/8} + C \quad \text{---} \quad (4)$$

continuing in this manner,

$$\frac{T(2)}{2} = \frac{T(1)}{1} + C \quad \text{---} \quad (5)$$

We add all the Eqn's from 1 to s and note that log n of them.

$$\frac{T(n)}{n} = \frac{T(1)}{1} + c \log n$$

which yields, $T(n) = cn \log n + n = O(n \log n)$

Average Case: The number of comparisons for first call on partition: Assume left to right moves over k smaller element and thus k comparisons. So when right to left crosses left to right it has made n-k+1 comparisons.

$T(n)$ = comparisons.

+

.

$$\{\Sigma\ 1 <= n_{left}, n_{right} <= n[T(n_{left}) + T(n_{right})]\}/n$$

$$= (n+1) + 2[T(0) + T(1) + T(2) + \ldots T(n-1)]/n$$

$$nT(n) = n(n+1) + 2[T(0) + \ldots + T(n-2) + T(n-1)]$$

$$(n-1)T(n-1) = (n-1)n + 2[T(0) + \ldots + T(n-2)]$$

Subtracting both the sides

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

Recurrence relation obtained is

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

using method of substitution

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

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

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

$T(n-2)/(n-1) = 2/(n-1) + T(n-3)/(n-2)$

$T(n-3)/(n-2) = 2/(n-2) + T(n-4)/(n-3)$

$\vdots$

$T(2)/3 = 2/3 + T(1)/2 \quad T(1)/2 = 2/2 + T(0)$

$T(n) = (n+1)\,2[\sum_{2 \le k \le n+1} 1/k$

$= 2(n+1)\,[\sum_{2 \le k \le n+1} 1/k]$

$= 2(n+1)[\log(n+1) - \log 2]$

$T(n) = O(n \log n)$

Worst case:

$T(n) = T(n-1) + c, n > 1$

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

$T(n-2) = T(n-3) + c(n-2)$

$\vdots$

$T(2) = T(1) + c(2)$

Adding up all these equations yields.

$T(n) = T(1) + \sum_{i=2}^{n} i$

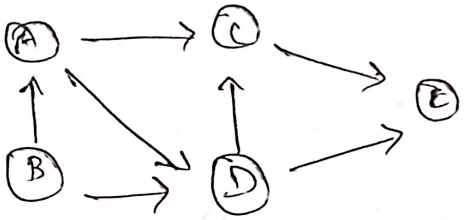
$= O(n^2)$

1b. Based on the principle of DAG, specific ordering of vertices is possible. This method of arranging the vertices in some specific manner called topological sort.

There are two commonly used algorithm for sorting the vertices using topological sort method.

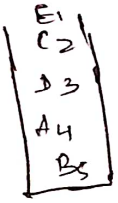
(a) DFS Based Algorithm:

Ex:



Solution:- as the graph is DAG, the topological sort is possible.

Step 1:- Find DFS and push the visited vertices in the stack



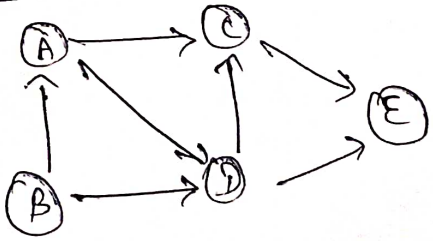
DFS stack.

Step 2:- Now pop-off the contents of the stack, E,C,D,A,B

Step 3:- Reverse the popped contents. The list which is gotten is a topologically sorted list.

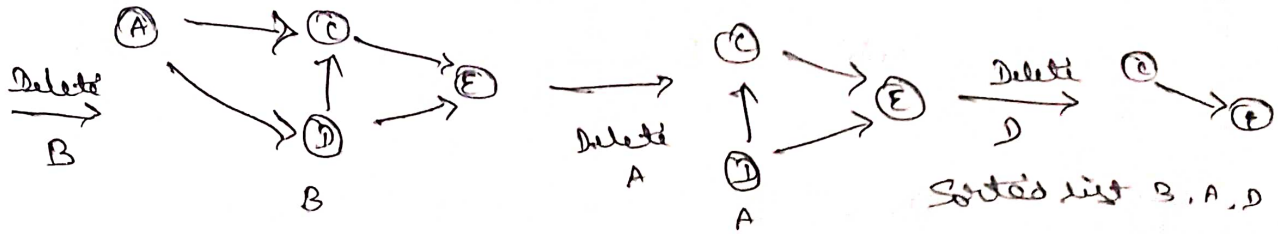
∴ B, A, D, C, E.

(b) Source removal Algorithm:-



Solution :- we will follow following steps to obtain topologically sorted list.

Choose vertex B, because it has no incoming Edge, delete it along with its adjacent edges.

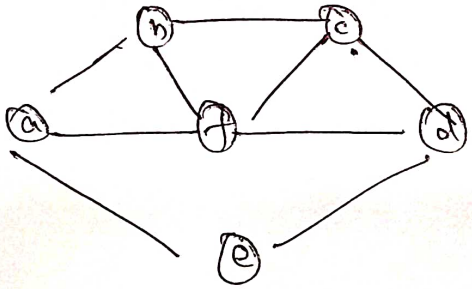


B, A, D, C B, A, D, C E

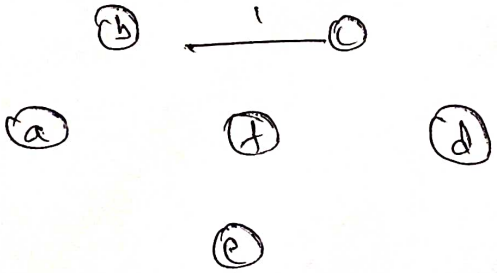
∴ list after topological sorting will be B, A, D, C, E.

(4a)

given :-

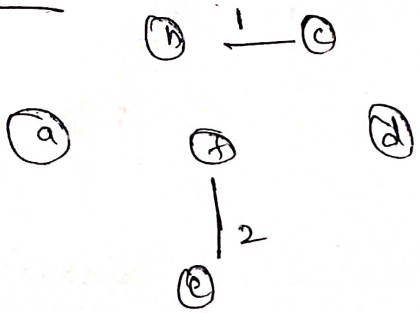


Step 1 :



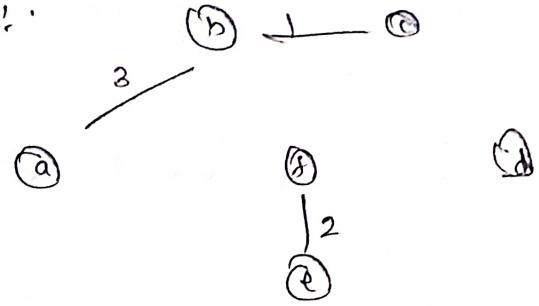
total wt = 1

Step 2 :-



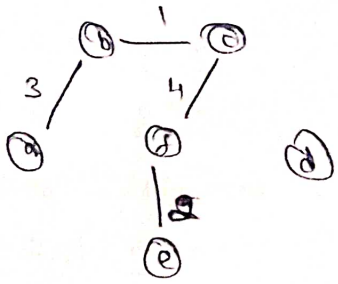
total wt = 3

Step 3!:



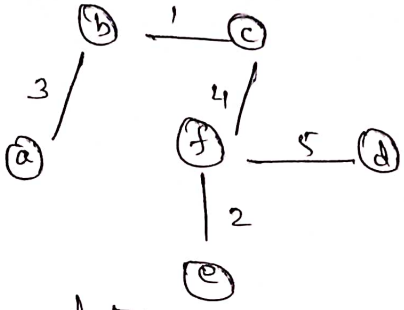
total wt = 6

Step 4!:



total wt = 10

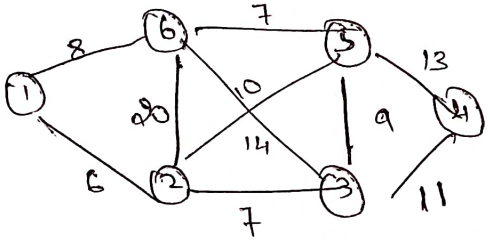
Step 5 !:



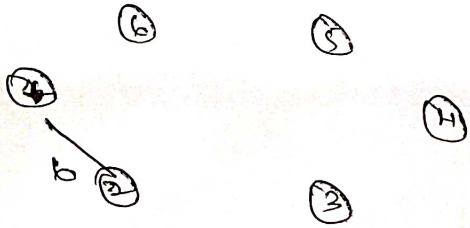
total wt = 15

The total cost of above minimum spanning tree is 15 //.

4 b.

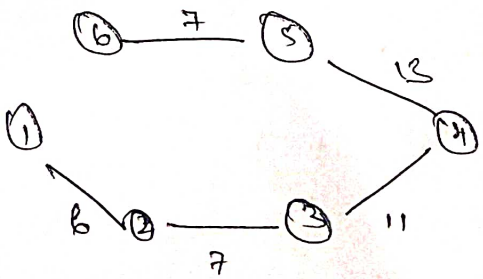


Step 1 !:

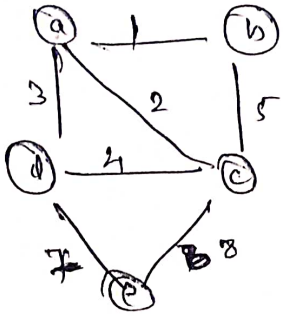


Step 2 !:

Similarly!:

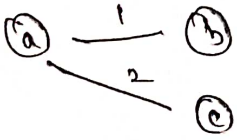


wt = 44 //.

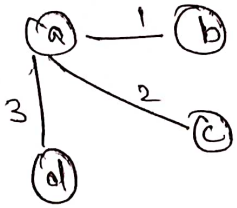


Step 1 :- ~~Start~~ select a-b

Step 2 :- Select a-c



Step 3 :- Select a-d



Step 4 :- Select d-e