Lecture 18

Quiz 1

Ques 1. A `nice' set for an undirected graph G=(V,E) is a set S of vertices such that for each $v \in V$, either v lies in S or a neighbor of v lies in S.
Devise the most efficient algorithm to compute a nice set of minimum possible size for an input tree T, and justify its correctness.

Solution.
Root T at an arbitrary vertex.
Initialize the set C (*the set of vertices to be covered*) as vertex-set of T, and invoke **Cover(C,T).**

ALGORITHM Cover(C,T):

1. If {C is empty} then return $\emptyset$.
2. $x \leftarrow$ A node in C having maximum possible depth.
3. $y \leftarrow$ parent(x, T)
4. $C' \leftarrow$ set obtained from C by removing from it y and neighbors of y.
5. Return $\{y\} \cup \textbf{Cover}\,(C', T)$.

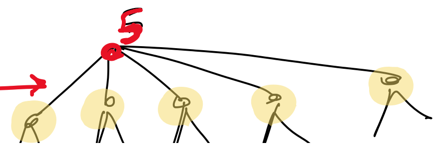
Correctness
Take an instance of **Cover(C,T).**

Exchange Lemma: Let x be a node in C of maximum depth, and y be parent(x, T). If S' is an optimal solution to cover C, then the set $S = (S' \setminus x) \cup y$ is also opt solution.

Take a solution S containing y. Note $(S \setminus y)$ must be an optimal solution to C' = C $\setminus \{N(y) \cup y\}$.
This proves the optimality of our algorithm.

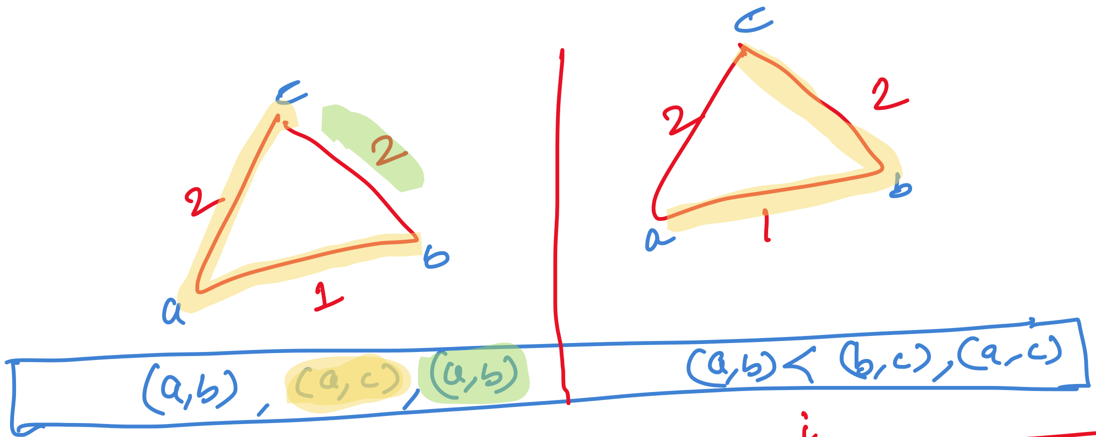
(Note: The algorithm can be implemented in linear-time by scanning vertices in T in bottom-up manner).

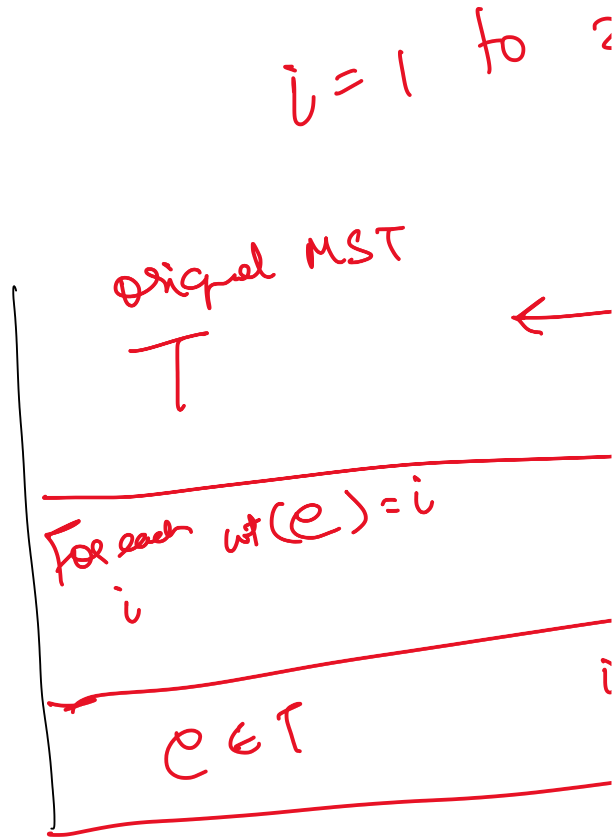
Wrong Solutions:
- Going by maximum degree
- Odd even levels



Ques 2. Given an MST of G with edge weights in range $[0,\ 2n]$. Find that sorted ordering of vertices over which Kruskal's algorithm will produce the given MST.

Solution.



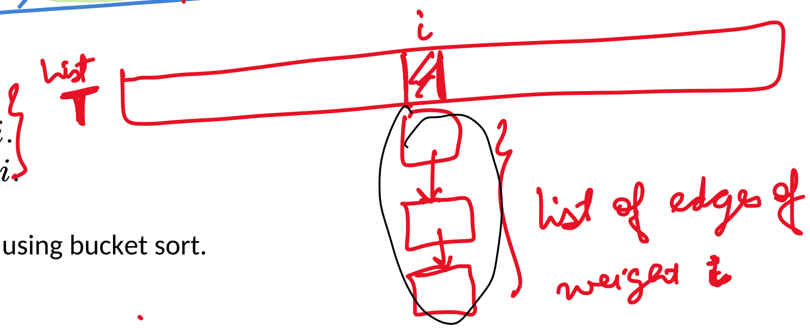


ALGORITHM

For $i = 1$ to $2n$:

Create two link-lists:

- $T[i]$ = to store edges of T having weight i.
- $G[i]$ = to store edges of G having weight i.



These lists can be created in $O(m+n)$ time using bucket sort.

1. Unmark all edges of G.
2. $L \leftarrow \emptyset$
3. For $i = 1$ to $2n$:
   i. For each $e \in T[i]$:
      -- L.append(e)
      -- Mark edge e
   ii. For each $e \in G[i]$:
      If e is unmarked:
         -- L.append(e)

Ques 3. Let G=(V,E) be a weighted graph with non-negative edge costs $c_e \geq 0$. Let T be an MST of G, and P be a shortest path from a vertex s to a vertex t.
Now suppose that the cost of each edge e of G is replaced by $(\boldsymbol{a}\, c_e - \boldsymbol{b})$, for some integers $a, b \in (-\infty, \infty)$.
Which of the following is true about the updated graph?

a = 2
b = 1
c = 0.0001

I.	If $a > 1$, then T remains an MST, and $\boxed{P \text{ remains an } s-t \text{ shortest path}}$.
II.	If all edge weights are the same and $a > b^4$, then T remains an MST, and P remains an $s-t$ shortest path.
III.	If $a < -1$, then T can never be an MST, and P can never be an s-t shortest path.
IV.	If $a = b = \lvert E \rvert$, then T remains an MST, but P need not be an s-t shortest path.
V.	If all edge weights are the same and $a > b^5$, then T remains an MST, and P remains an $s-t$ shortest path.

correct

1. Wrong -- P needn't be shortest path
   (one $s-t$ path with direct edge of weight 2, one 2-hop path with weight 1, $b = 100$).

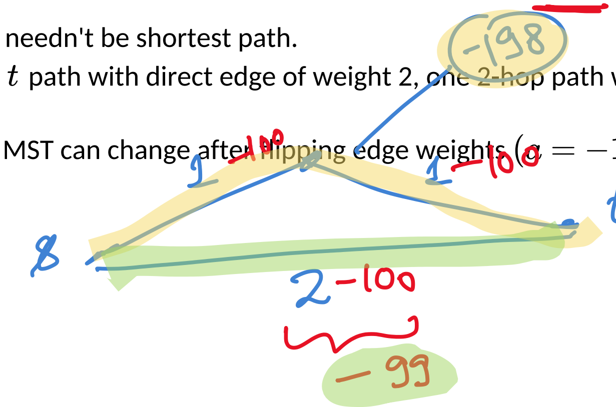
2. Wrong -- A long path can now become shorter on adding $-b$ to all the edges.

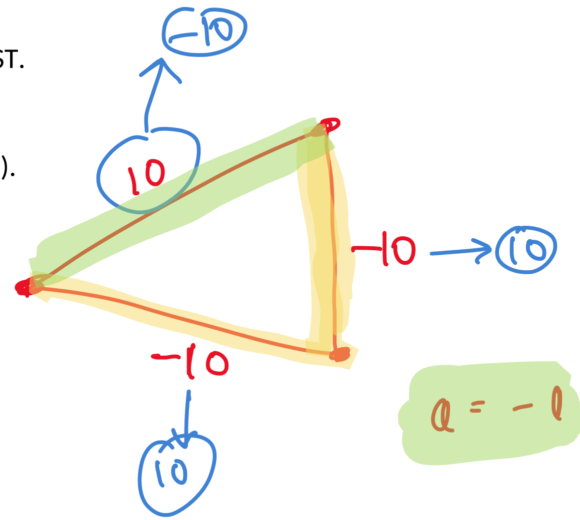
3. Wrong -- If all edge weights were earlier zero, then any original MST will remain an MST.

4. Right -- P needn't be shortest path.
   (one $s-t$ path with direct edge of weight 2, one 2-hop path with weight 1, $b = 100$).

b=100

5. Wrong -- MST can change after flipping edge weights $(a = -1,\ b = 0)$.



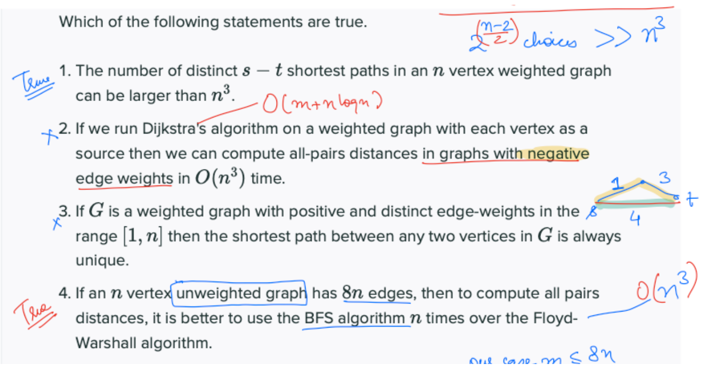


Ques 4. Which of the following statements holds for Huffman's encoding scheme with n symbols?

Select one or more:

I.	A letter with a frequency larger than 51% might get encoded with less than two bits.
II.	If n is a power of 2, then the encoding of each symbol must be bounded by 2 $log_2\ n$.
III.	If all frequencies are less than 26%, then all letters will be coded with two or more bits.
IV.	In optimal coding, a letter with a frequency of at least 66% is always encoded with one bit.
V.	All the least frequency symbols must have identical length codes for the encoding to be optimal.
VI.	The most frequent letter must always be encoded as a one-bit code.

Quiz 2



Which of the following statements are true.

$2^{(\frac{n-2}{2})}$ choices $>> n^3$

True 1. The number of distinct $s-t$ shortest paths in an n vertex weighted graph can be larger than n^3.

$O(m + n \log n)$

X 2. If we run Dijkstra's algorithm on a weighted graph with each vertex as a source then we can compute all-pairs distances in graphs with negative edge weights in $O(n^3)$ time.

X 3. If G is a weighted graph with positive and distinct edge-weights in the range $[1, n]$ then the shortest path between any two vertices in G is always unique.

True 4. If an n vertex $\boxed{\text{unweighted graph}}$ has $8n$ edges, then to compute all pairs distances, it is better to use the BFS algorithm n times over the Floyd-Warshall algorithm.

$O(n^3)$

our case $m \le 8n$

Write all the correct options below:

answers

(NOTE: There is Negative marking for wrong answers).

Time (BFS) $= O(m+n)$
$= O(n)$

BFS from all vertex as source $= O(n^2)$

Q2 DP on Strings (Similar to LCS)

15 Points

LCS

S present in A

S prest in B

Let A = $(a_1, ..., a_m)$ and B = $(b_1, ..., b_n)$ be two sequences. Your goal is to find the length of a smallest possible sequence, say C, such that both A and B are subsequences of C.

Smallest super sequence.

Complete the pseudo-code below to obtain a valid algorithm.
[5 × 3 = 15 marks]

one way $\rightarrow (m+n) - |LCS(A,B)|$

Main:

Create a 2D array Q of size $(m+1) \times (n+1)$ with starting index as (0, 0)

For $i = 0$ to m do $Q[i, 0] =$ [i]

For $j = 0$ to n do $Q[0, j] =$ [j]

$a_1 \; .. \; a_{i-1} \; a_i$

$b_1 \; ... \; b_{j-1} \; b_j$

For $i = 1$ to m
    For $j = 1$ to n
        If $(a_i = b_j)$ then $Q[i, j] =$ [$1 + Q[i-1, j-1]$]
        Else $Q[i, j] = \min($[____], [____]$)$

$\{ 1 + Q[i, j-1], \; 1 + Q[i-1, j] \}$

Return $Q[m, n]$

---

answer

answer

answer

answer

answer

Q3 DFS and Its Application

15 Points

Let G be a connected undirected graph. Fill in the blanks below to obtain an algorithm to compute all the bridge edges (x, y) in G for which both endpoints x, y are cut vertices.

[5 × 3 = 15 marks]

DFS(x) min {hp(x), hp(y)}

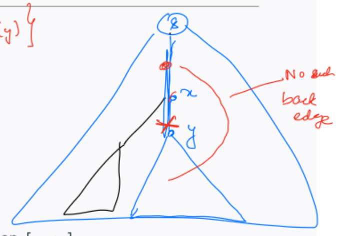
```
visited[x] = True
high-point[x] = level[x]
For Each y ∈ N(x) :
    If ( visited[y] = False ) then
        level[y] = 1 + level[x]
        Invoke DFS(y)
        high-point[x] = [____]
        If ( [____] & [____] & [____] ) then [____]
    If ( level[y] ≤ level[x] - 2 ) then
        high-point[x] = min( level[y], high-point[x] )
```

Bridge edge.

high point[y] = level(y)

L. append (x, y)

$|N(x)| > 1$ $|N(y)| > 1$



Main:

```
Let L be an empty list, and let s be an arbitrary vertex in G
For each u ∈ V(G) : set visited[u] = False
level[s] = 0
Invoke DFS(s)
Return L
```

---

answer

answer

answer

answer

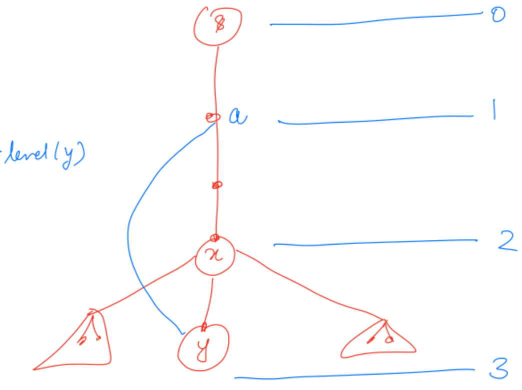
answer

$\boxed{|N(y)| > 1, \ |N(x)| > 1}$

Those bridge edges (x, y) such that x, y are cut vertices.

Removing "y" doesn't ... from connectivity if deg(y) = 1

high part(y) $\neq$ level(y)



x - root
x - one child

y is leaf