CSci 242: Algorithms and Data Structures

Instructor: Dr. M. E. Kim Date: December 13th, 2018

# Final Exam: Solution

**Q1.** [30] **True or False**

Justify your answer clearly. Give a counter example if necessary.

**(1)** [2+3] In the Kruskal's algorithm to find the Minimum Spanning Tree, the Priority Queue maintains the vertices of graph in order to get the vertex with the minimum weight edge.

False. PQ in Kruskal's algorithm maintains an weighted **edge** to remove/choose the minimum weight edge.

**(2)** [2+3] A large *minimum degree B* of B-tree, where B is the number of items that can fit in one block, increases the number of disk access required to find a key.

False. A large maximum degree b increases a branching factor of B-tree and it reduces the height of B-tree which is the maximum number of disk access.
Thus, the number of disk access required to find a key is decreased.

**(3)** [2+3] Quick sort is much faster than quadratic sorting algorithm such as insertion-sort and selection-sort.

False. Quick sort runs in $O(n^2)$ in the worst case, not faster than but as fast as the quadratic sorting algorithms such as insertion-sort and selection-sort.

However, it runs in $O(n \log n)$ in the best or average case so that it runs faster than quadratic sorting algorithms.

**(4)** [2+3] Dijkstra's algorithm is successful to find the shortest path from a given source in the graph which contains a negative weight edge.

False, Dijkstra's algorithm fails to find the shortest path in the graph with a negative weight edge.
It goes to the infinite loop in such a case.

**(5)** [2+3] The optimal data encoding/decoding can be *always* achieved by a variable length code word with no ambiguity.

False. The code word should be a prefix code to prevent from the ambiguity in decoding as well as a variable length code for the optimality.

**(6)** [2+3] A greedy algorithm makes a greedy choice before finding an optimal solution of sub- problem. Thus, it solves a problem in top-down approach.

True, A greedy choice is made prior to solving the subproblem.
Thus, it solves a problem in top-down approach.

**Q2.** [40] **Short Answer**

Explain or define your answer clearly.

(1) [5] Define a (*a, b*)-*tree* and explain its property.

- An **(*a, b*) tree** is a multiway search tree T with the following additional restrictions, where parameters *a* and *b* are integers such that $2 \le a \le (b+1)/2$:
- **Size Property**: Each internal node has at least *a* children, unless it is the root, and has at most *b* children.
  A root has at least 2 children because it stores at least one item.
- **Depth Property**: All the external nodes have the *same depth*.

(2) [5] Give the maximum height and the minimum height of (*a, b*)-tree in the asymptotic notation O and Ω, respectively.

$$\Omega(\log_b n) \le h \le O(\log_a n)$$

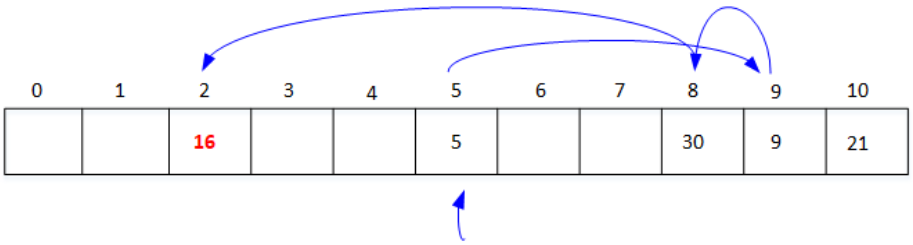
(3) [5] For a B-tree T of the maximum degree *b* and the minimum degree $a=\lceil b/2 \rceil$, what is (A) the minimum number of items and (B) the maximum number of items that a root can store, respectively?

Minimum number of key = 1 and maximum number of keys = b-1

(4) [5] Define a strongly connected component of a graph G = (V, E).

A strongly connected component of a graph is a maximal connected subgraph of G or a maximal subset of vertices of G in which every vertex is reachable from/to all other vertices.

(5) [10] For a given hash table of size 11, insert the keys 16 where a collision is handled by ***quadratic probing***.
The main hash function is ***h(k)=k* mod 11** and the **secondary quadratic function *f(j)* = $3j^2 + j$.**
Show the proper computational steps.




H(k, j) = ( h(k) + f(j) ) mod 11

H(16, 0) = 16 mod 11 = 5 --- collision with 5 at A[5].

H(16, 1) = (5 + (3+1)) mod 11 = 9 – collision with 9 at A[9].

H(16, 2) = (5 + (3*4+2)) mod 11 = 19 mod 11 = 8 -- collision with 30 at A[8].

H(16, 3) = (5 + (3*9+3)) mod 11 = 35 mod 11 = 2

So, 16 is inserted at A[2].

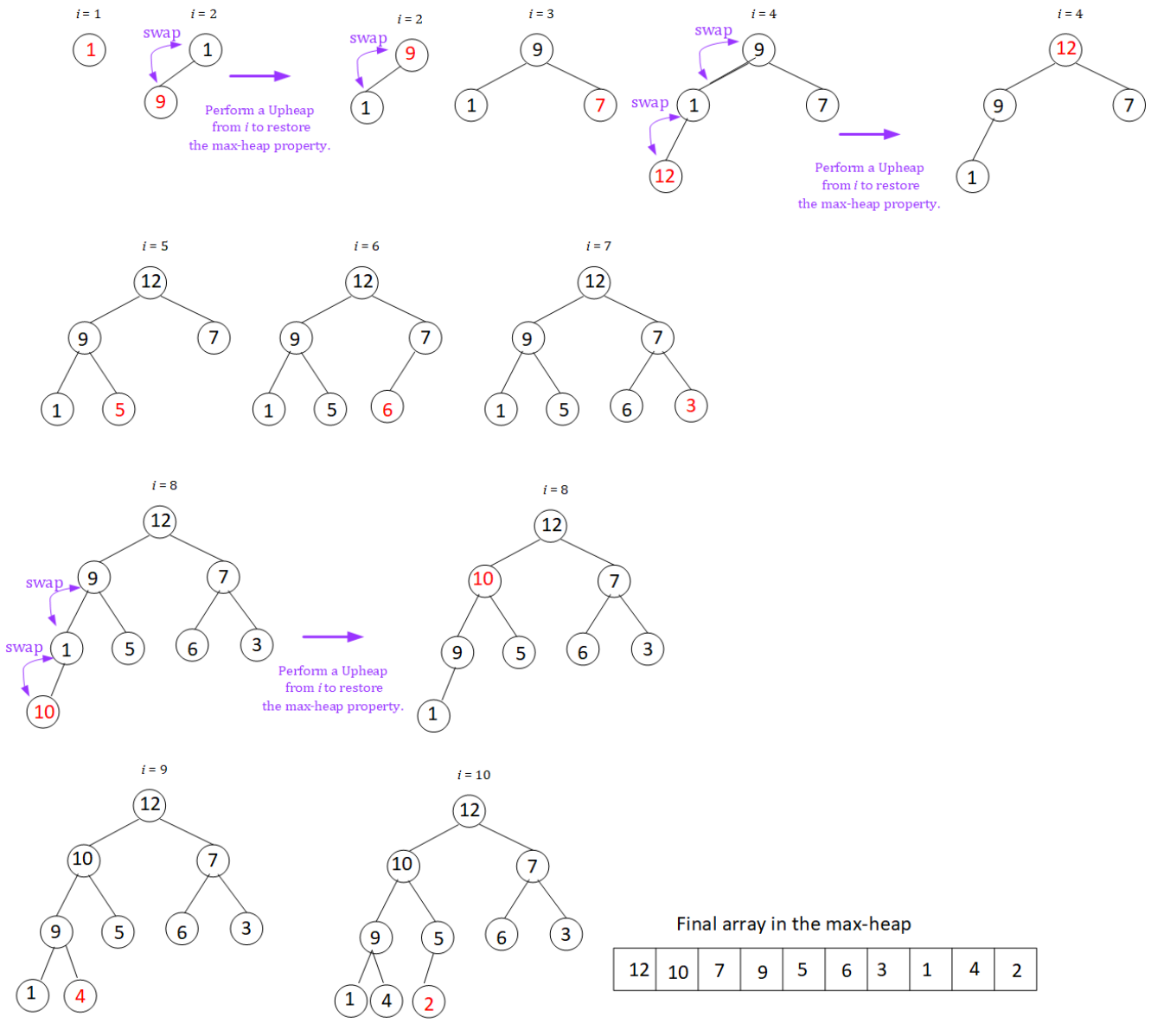
(6) [5] Describe 4 types of edge in Depth First Search in Directed Graph.

For an edge v → w where v is active

- discovery edges: forms a DFS tree
  - *w* is unexplored and not active
- back edges: to ancestor vertex *w*
  - *w* is unexplored but active
  - *v* is a descendant of *w*
- forward edges: to a descendant vertex *w*
  - *w* is explored and not active and is a descendant of *v*
- cross edges: neither to ancestor nor to descendant
  - *w* is explored but neither a descendant of *v* nor an ancestor of *v*.
  - Can go between vertices in same DFS tree or in different DFS trees.

(7) [5] For a tree stored in the given array, build a *maximum* heap. Show the final heap in the array. Do not use 'Bottom-Heap' construction, but by performing 'UpHeap' operation from *j* = 1 to *n*.

| 1 | 9 | 7 | 12 | 5 | 6 | 3 | 10 | 4 | 2 |
|---|---|---|---|---|---|---|---|---|---|




Final array in the max-heap

| 12 | 10 | 7 | 9 | 5 | 6 | 3 | 1 | 4 | 2 |
|---|---|---|---|---|---|---|---|---|---|

**Q3. [20] Fractional Knapsack Problem**

An edited book has 8 articles. The table shows the lengths of the articles and their importance, where the scale of importance is 1(low) to 10(high). The book must be at most 200 pages long. The problem is ***to edit the book*** so that **the overall importance is maximized.**

(1) [15] Edit the book by choosing articles whose pages and importance are given in the table, giving

| Article | Importance of article | Pages | Unit importance | Rank of unit importance |
|---|---|---|---|---|
| A | 3 | 15 | 1/5 = .2 | 4 |
| B | 5 | 20 | 5/20 = .25 | 3 |
| C | 6 | 20 | 6/20 = .3 | 2 |
| D | 4 | 10 | 4/10 = .4 | 1 |
| E | 3 | 30 | 3/30 = .1 | 6 |
| F | 3 | 90 | 3/90 =1/30 = .033 | 8 |
| G | 4 | 60 | 4/60 =.066 | 7 |
| H | 6 | 40 | 6/40 = .15 | 5 |

(A) [5] the list of the chosen articles with their chosen number of pages in the order,

D/10 + C/20 + B/20 + A/15 + H/40 + E/30 + G/60 + F/5

(B) [5] the importance of each chosen article, and

D/4 + C/6 + B/5 + A/3 + H/6 + E/3 + G/4 + F/(1/6 =.166)

(C) [5] the total maximum importance of the edited book of 200 pages.

31.16

(2) [5] Consider a greedy rule for the above Fractional Knapsack Problem that selects the articles in non-decreasing order of pages. If the capacity of the knapsack is not exceeded, we take all of the pages. Otherwise, we take whatever portion of the article fills the knapsack and stop.
Give an example to show that this greedy algorithm does not necessarily maximize the importance.

If an article is chosen in non-decreasing order of pages, the order and the # of chosen pages are:

D/(10, 4) + A/(15, 3) + B/(20, 5) + C/(20, 6) + E/(30, 3) + H/(40, 6) + G/(60, 4) + F/(5, 5*3/90)

and the total importance is 31.16

Coincidentally, the importance is equal in this case.
However, if the edited book is 100 page, the importance by the original greedy choice in (1) is

D/(10,4) + C/(20, 6) + B/(20, 5) + A/(15, 3) + H/(35, 35*6/40) = 23.25

while the total importance by a choice in (2) is

D/(10, 4) + A/(15, 3) + B/(20, 5) + C/(20, 6) + E/(30, 3) + H/(5, 5*6/40) = 21.75 < 23.25.

Thus, this greedy choice doesn't necessarily maximize the importance.

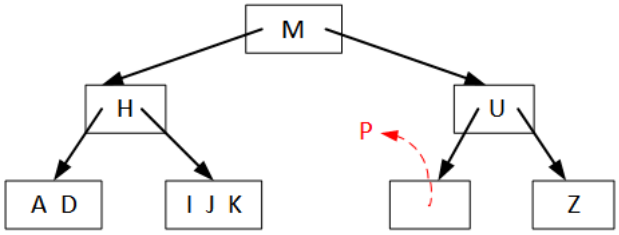
**Q4.** [20] **B-Tree**

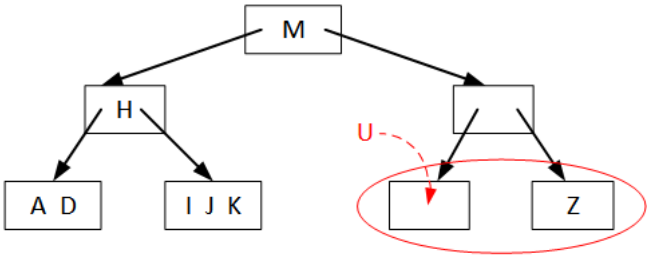
**(1)** [14] For a given B-tree whose order is 4, show the changes of B-tree

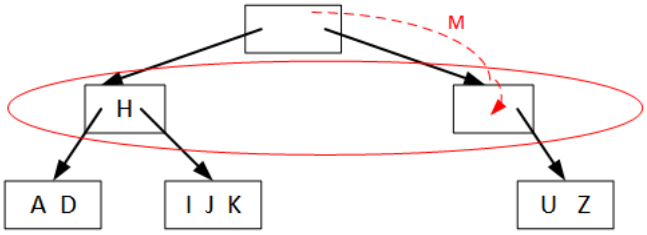
(A) after deleting a key ***P***, then,

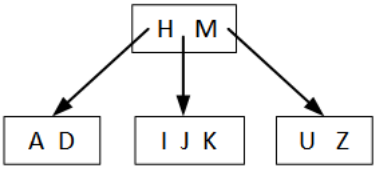
(B) inserting a key ***L*** to the B-tree in (A). Show the deletion & insertion step by step.

(A) delete P

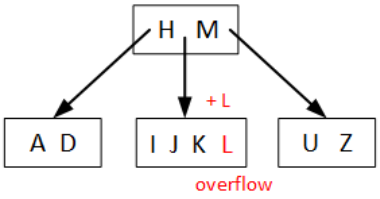


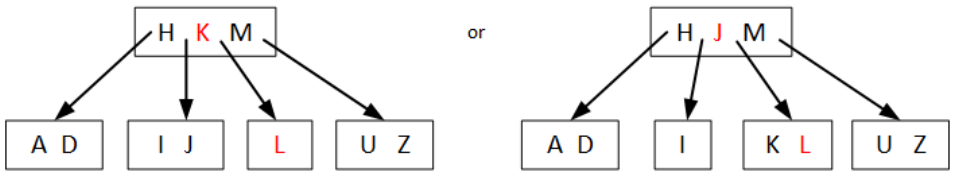




(B) insert L




Split a node and move a median key K (or J) to its parent →

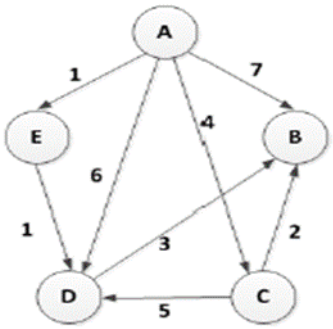


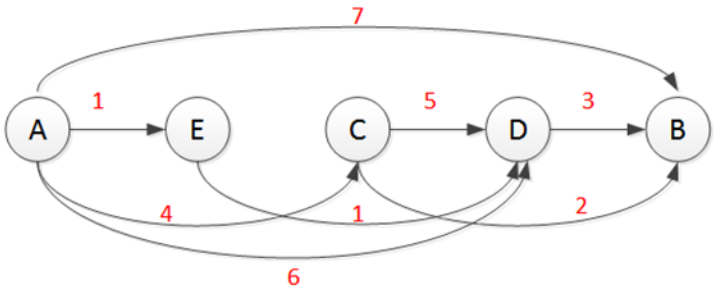
**(2)** [6] Describe 3 *page replacement strategies* in *blocking* if the primary memory is full when data is transferred from the external memory

- Random strategy: a randomly chosen page is evicted from the cache.
- FIFO strategy: evicts the page that has been in the cache the longest, i.e. the page that was transferred to the cache furthest in the past.
- LRU strategy: evicts the page that was least recently used.

**Q5.** [20] **Single Source Shortest Path in the DAG**

(1) [10] For a given DAG, (A) sort the vertices in the topological order and (B) redraw the graph by arranging the vertices in the sorted order

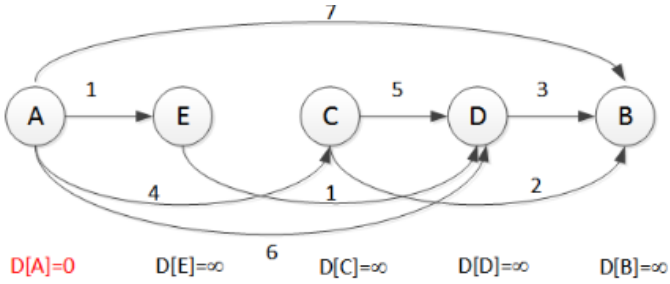


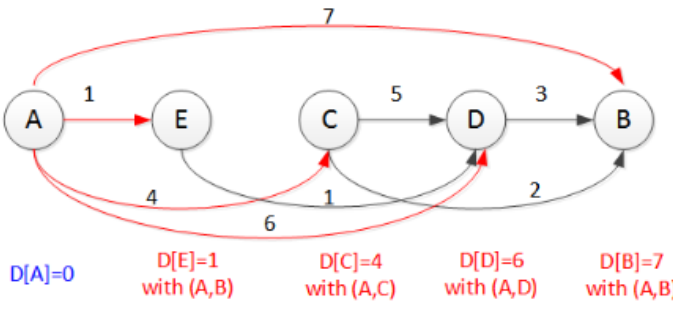

Vertices are in the topological order and DAG is redrawn.

A (E) C (E) D B

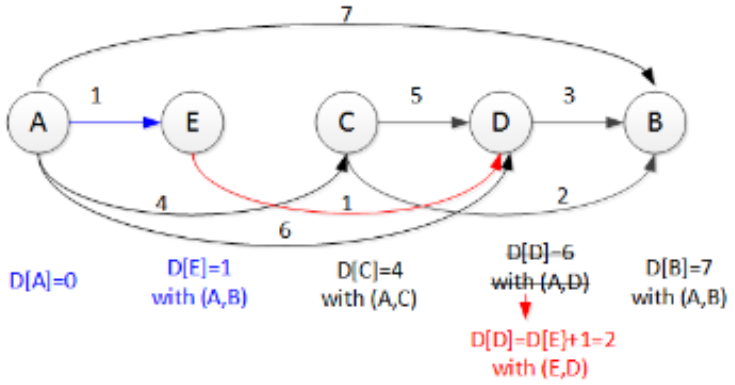
(2) [10] Find the shortest path from a vertex *A* to each vertex in (1). You have to show the proper steps of edge relaxations, updating a key, D[*v*] of each vertex *v*.



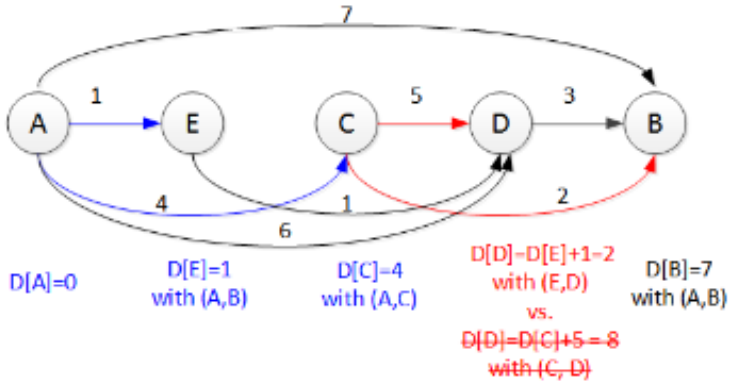

1. Relax the incident edges on A



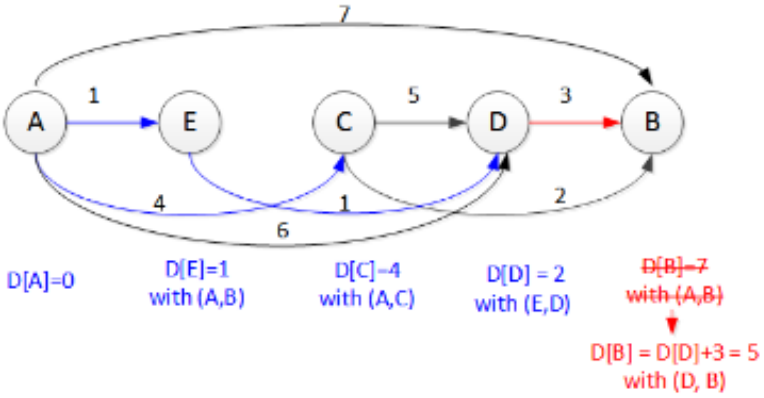
2. Relax the incident edges on E.



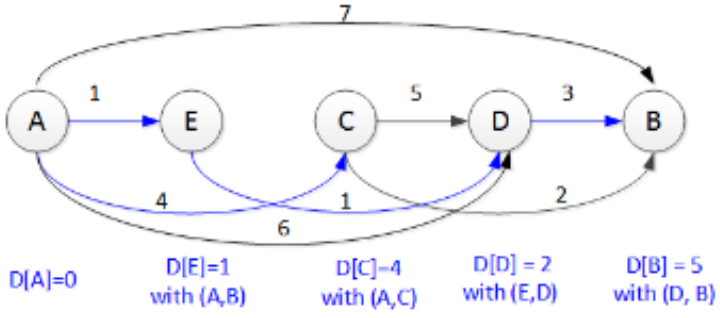

3. Relax the incident edges on C.




4. Relax the incident edges on D.




$n$-1 edges have been relaxed. Complete.

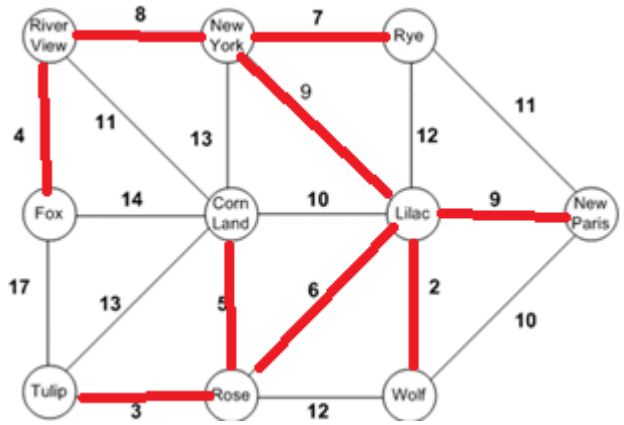


**Q6.** [20] **Minimum Spanning Tree (MST)**

(1) [10] Write Kruskal's algorithm in the pseudo code to find the *Minimum Spanning Tree*.

**Algorithm** KruskalMST($G$):
*Input:* A simple connected weighted graph $G$ with $n$ vertices and $m$ edges
*Output:* A minimum spanning tree $T$ for $G$
**for** each vertex $v$ in $G$ **do**
Define an elementary cluster $C(v) \leftarrow \{v\}$.
Let $Q$ be a priority queue storing the edges in $G$, using edge weights as keys
$T \leftarrow \emptyset$ // $T$ will ultimately contain the edges of the MST
**while** $T$ has fewer than $n-1$ edges **do**
$(u, v) \leftarrow Q$.removeMin()
Let $C(v)$ be the cluster containing $v$
Let $C(u)$ be the cluster containing $u$
**if** $C(v) \neq C(u)$ **then**
Add edge $(v, u)$ to $T$
Merge $C(v)$ and $C(u)$ into one cluster, that is, union $C(v)$ and $C(u)$
**return** tree $T$

(2) [10] By applying Kruskal's algorithm to the given graph, (A) find its MST and (B) give the *total minimum weight*. You should show proper *changes of clusters*, the MST in growing, the smallest weight edge that cross the clouds.




Initially, each vertex forms a single cluster: {RV}, {NY}, {Rye}, {Fox}, {CL}, {Lilac}, {NP}, {Tulip}, {Rome}, {Wolf}

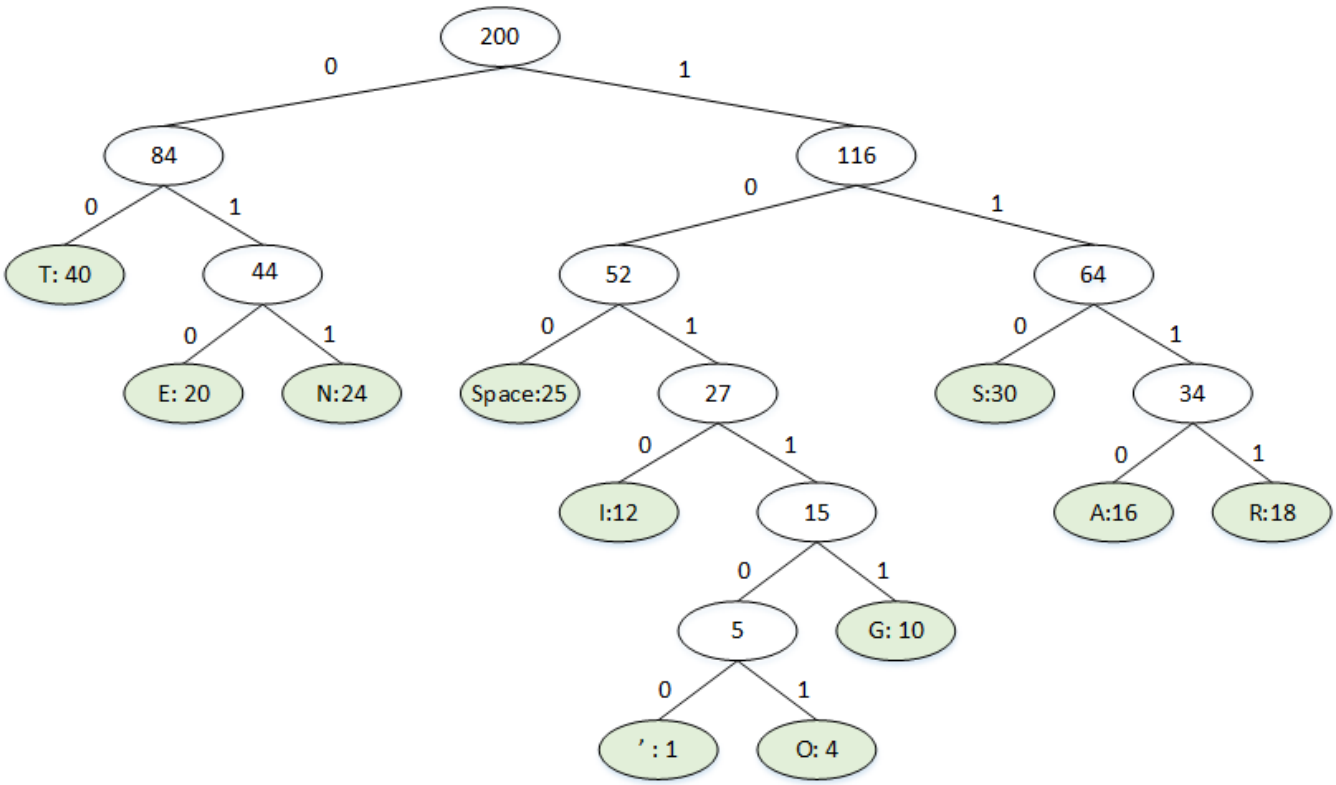
1. w(Lilac, Wofl) = 2 is chosen. Cluster: {Lilac, Wolf}, etc.
2. w(Tulip, Rome) = 3 is chosen Cluster: {Tulip, Rome}, {Lilac, Wolf}, etc.
3. w(RiverView, Fox) = 4. Cluster: { RV, Fox}, … etc.
4. w(ConLand, Rome) = 5. Cluster:{CL, Rome, Tulip}, …, etc.
5. w(Rome, Lilac) = 6. Cluster: {CL, Rome, Tulip, Lilac, Wolf}, etc.
6. w(NY, Rye) = 7. Cluster: {NY, Rye}, etc.
7. w(RiverView, NY) = 8 Cluster: {RV, NY, Fox, Rye}, etc.
8. w(NY, Lilac) = 9. Cluster:{Fox, RV, NY, Rye} + {Lilac, CL, Rome, Tulip, Wolf}
= {Fox, RV, NY, Rye, Lilac, CL, Rome, Tulip, Wolf}, {NP}
9. w(Lilac, NP) = 9. Cluster = G(V)

Total weight of MST = 53

**Q7.** [20, Optional] **Huffman Codes**

**(1)** [10] The text contains only the characters in the table with the given frequency. (A) Construct a Huffman Tree to generate and (B) give the optimal Huffman code for each character.

| Character | Frequency | Huffman Code |
|---|---|---|
| Space | 25 | 100 |
| A | 16 | 1110 |
| E | 20 | 010 |
| I | 12 | 1010 |
| O | 4 | 101101 |
| G | 10 | 10111 |
| N | 24 | 011 |
| R | 18 | 1111 |
| S | 30 | 110 |
| T | 40 | 00 |
| ’ | 1 | 101100 |




**(2)** [5] What is the total number of bits required to encode the text using your Huffman codes?

$\sum_{i='}^{space} |code_i| = 641$

**(3)** [5] Decode the following codes into a text.

110 010 1110 110 101101 011 101100 110 100 10111 1111 010 010 00 1010 011 10111 110

S E A S O N ' S _ G R E E T I N G S

**Q8.** [10, Optional] **Recurrence**

Solve the following recurrence by one of the following methods:
Master's Theorem, Iterative Substitution or Recursion Tree method.

$$T(n) = \begin{cases} O(1) & n < 4 \\ 3 \cdot T\left(\frac{n}{4}\right) + O(n) & n \geq 4 \end{cases}$$

You should clearly state the followings, depending of the method of your choice

- Master's Theorem :
  the case to which it belongs, the rationale of the solution and the solution in the asymptotic bound,

- Iterative Substitution:
  The proper number of iterative steps and the final step in which the number of input is 1, the computation of some parameters.

- Recursion Tree :
  the height of tree, the number of leaves, level, the number of nodes per level, the size of input per level, a per-level time, the total time **and** its asymptotic bound (i.e. the asymptotic bound of solution),

By Master's Theorem:
a = 3 and b = 4, f(n) = O(n) = $cn^1$
$\log_4 3 < 1$
Since f(n) = O(n1) = O($n^{\log_3 4+\varepsilon}$), i.e. $n^{\log_3 4} < f(n) = O(n)$, and af(n/b)= 3 ·n/4 ≤ .75* f(n),
it belongs to the case 3. So, T(n) = Θ(f(n)).

Therefore, T(n) = Θ(n)