CS/ECE 374, Fall 2018 Midterm 2: Problem 1	Gradescope name: Xinrui Ying

Short questions. No justification is required for your answers.

- Give an asymptotically tight bound for the following recurrence.

$$T(n) = T(n-2) + n^2 \quad n \geq 3 \text{ and } T(n) = 1 \quad 1 \leq n \leq 2.$$

n^2

$(n-2)^2$

$(n-4)^2$

$(n-6)^2$

$\Theta(n^2)$

- Suppose we have k numbers $a_1, a_2, \ldots, a_k$ each of which is binary number with at most h digits.
  - Give an asymptotic upper bound on the number of digits required for the product $a_1 a_2 \ldots a_k$ as a function of k and h.

The answer: $O(h + \lfloor \log_2 k \rfloor)$

h h h h

h+1 h+1

h+2

  - Give an asymptotic upper bound on the number of digits required for the number 2^{a_1} as a function of h.

The answer: $O(2^{h+1})$

$a_1 \rightarrow$ h digits

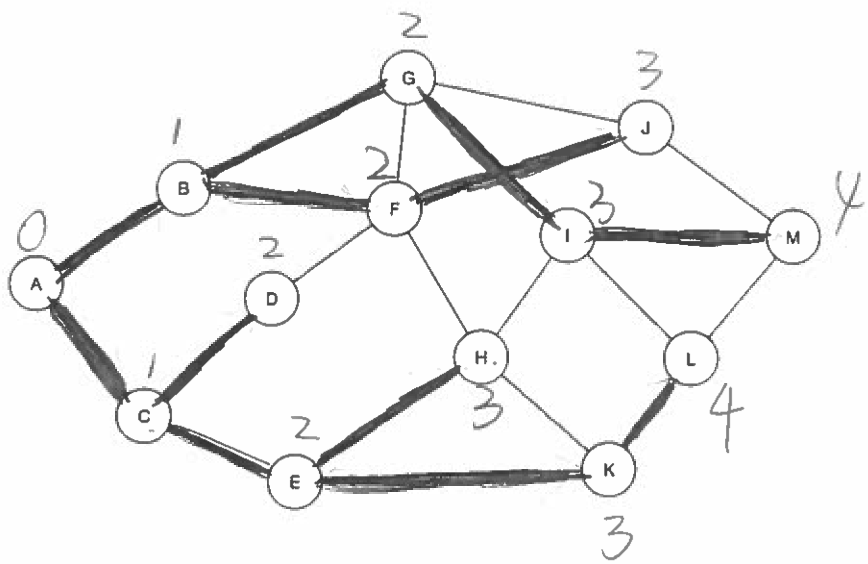
1111 11 11

$\boxed{2^{h+1}}$

CS/ECE 374, Fall 2018 Midterm 2: Problem 2a	Gradescope name: Xinrui Ying

(a) In the graph shown in the figure below, draw a BFS tree rooted at node *A* and indicate the distances of each node from *A*. You can indicate the tree edges by drawing over the given figure. Make sure it is easy to see which edges are part of the tree.

See next page for part (b).



CS/ECE 374, Fall 2018	Gradescope name:
Midterm 2: Problem 2b	Xinrui Ying

(b) Let T be a BFS tree for a connected undirected graph $G = (V, E)$ rooted at s (that is, BFS was started at s). Suppose there are two distinct nodes u, v that are the same distance from s and (u, v) is an edge. Prove that there is an odd-length cycle C in G that contains the edge (u, v).

Since u, v are the same distance from s, there will be a node that is there ancestor, and none of their paths from the root should contains the other node, which mean the path (s->u) should not contain v and the path (s->v) should not contain u. Othewise, the distance of s->v and s->u will not be the same. From the latest common ancestor, the vertex in these two paths will be disjoint (not including the ancestor).

Let the ancestor be k, we claim that k->u and k->v should be the same. Since there is only one path s->k in a tree, one path from s->u and one path s->v, s->k and k->v should be s->v, same for s->k and k->u equals s->u. Since $d(s,v) = d(s,u)$, $d(s,k)+d(k,u) = d(s,k)+d(k,v)$ $d(k,u) = d(k,v)$. Combine the three path k to u, k to v and the edge between uv. The length of cycle will be $d(k,v)+d(k,u)+1 = 2d(k,v)+1$

We proved there is a cycle contains uv and the length of cycle is odd.

Since $d(k,v) \in N$, then $2d(k,v)$ is even thus $2d(k,v)+1$ is odd.

CS/ECE 374, Fall 2018 Midterm 2: Problem 3a	Gradescope name: Xinrui Ying

Given a graph $G = (V, E)$ a vertex cover of G is a subset $S \subseteq V$ of vertices such that for every edge $(u, v) \in E$, u or v is in S. The goal in the minimum vertex cover problem is to find a vertex cover S of smallest size. In the weighted version of the problem, vertices have non-negative weights $w : V \to \mathbb{Z}_+$, and the goal is to find a vertex cover of minimum weight.

(a) Describe an efficient algorithm for the minimum weight vertex cover problem in a given tree $T = (V, E)$. Your algorithm only needs to compute the weight. See next page for a figure which may help you.

Since it's a tree, there is a root, starting from the root, at each vertex, we have two choices:

1. take the vertex and don't take any of it's children.
2. Take all it's children and don't take the vertex.

Both way will cover all the edge from the vertex.

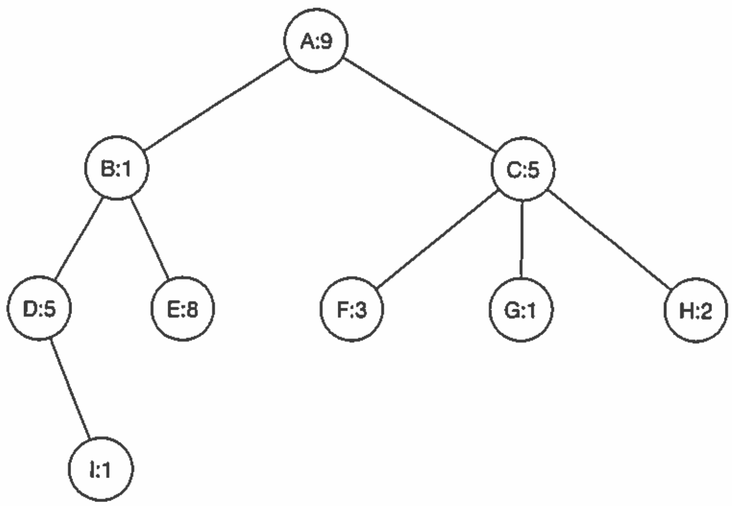
The function minwei(V, i) takes a vertex and a int i, returns min weight of vertex cover

$$
\text{minwei}(v,i)\begin{cases} 0 & V = \text{null} \\ \text{MIN}\{w(v) + \sum_{u \in N(v)} \text{minwei}(u,0),\ \sum_{u \in N(v)} \text{minwei}(u,1)\} & i=0,\ V \neq \text{null} \\ \sum_{u \in N(v)} \text{minwei}(u,0) + w(v) & i=1,\ V \neq \text{null} \end{cases}
$$

If i=0 means you can choose to have take the vertex. if i=1, means the other endpoint of the edges it connects to it's parent it's not selected. Find the source in graph T, and do it on the Source minwei(source, 0). Since each node will have 2 sithation: must chooce or can choose. The Space it's O(2n) and time will be O(n).

CS/ECE 374, Fall 2018 Midterm 2: Problem 3b	Gradescope name: Xinrui Ying

(b) Compute the minimum weight vertex cover in the tree shown in the figure below. The weights are shown next to the label of the node.



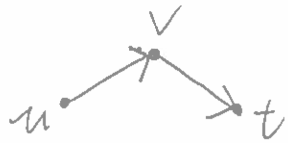
{B, I, C} → minimum vertex set

weight = 7.

CS/ECE 374, Fall 2018 Midterm 2: Problem 4a	Gradescope name: Xinrui Ying

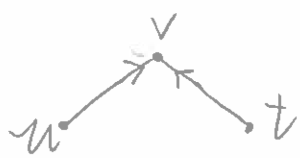
- Draw an example of a small DAG G such that adding a *single* edge to G makes it strongly connected.
- Draw an example of a small DAG G such that adding any one edge does not make it strongly connected.

See next page for the second part.



→ graph for the first part.

adding a edge (t,u) will make the graph strongly connected.



→ graph for second part

v is a sink and there is no edge can be added to v.

CS/ECE 374, Fall 2018	Gradescope name:
Midterm 2: Problem 4b	Xinrui Ying

- Describe an efficient algorithm that when given a DAG $G = (V, E)$ checks whether a single edge can be added to G to make it strongly connected.
- Suppose you are now given a directed graph $G = (V, E)$ (which may or may not be a DAG). Describe an efficient algorithm that checks whether one can add at most one edge to make G strongly connected.

For both parts the ideal running time is linear in the graph size but slower correct algorithms will get partial credit.

For DAG $G=(V,E)$

Find source of G first, if there are multiple source, return false. find the sink as well, if there are multiple sinks, return false. If there is exactly one sink and one source, create an edge from sink to source and find a strongly connected component in new graph using DFS on any one vertex. If the number of vertex in the strongly connected component is equal to $|V|$, return true, else false. Finding source, sink and adding an edge are all in linear time, do DFS to find strongly connected components is linear time, so total in linear time.

For directed graph $G=(V,E)$.

Do DFS on G to find all strongly connected components and build graph G^{SCC}. If there is only one vertex in G^{SCC}, return true, else do above algorithm on G^{SCC}. The running time is still linear as DFS take linear and build G^{SCC} takes linear time.

CS/ECE 374, Fall 2018	Gradescope name:
Midterm 2: Problem 5	Xinrui Ying

- Let B and C be two *sorted* arrays of integers of length n and m, respectively. Assume that all the numbers in the arrays are distinct. Describe an $O(n+m)$ algorithm to count the number of pairs $(b,c) \in B \times C$ such that $b > c$. For example, if $B = [5,11,16]$ and $C = [9,14]$, there are three such pairs: $(11,9),(16,9),(16,14)$ and hence the output of the algorithm should be 3. Note that the algorithm only needs to output the count and not the actual pairs.
- Suppose A is an *unsorted* array of numbers with all numbers distinct. Then $(A[i],A[j])$ is an inversion if $i < j$ and $A[i] > A[j]$. For example if $A = [11,5,12,10]$ then the inversions are $(11,5),(11,10),(12,10)$. Given an array A of n integers describe an efficient algorithm to count the number of inversions in A. *Hint:* Modify MergeSort and use the preceding part.

For part 1:

Set i=1, j=1, count = 0 at first, compare B[i] and C[j], if B[i] > C[j], just do j+1. If B[i] < C[j], adding (j-1) to the count. If i is bigger than number of elements in B, return count. If j is bigger than number of elements in C, Count = count + (|B|-i+1) × (j-1) and return it. The algorithm takes linear time O(m+n) since we traverse each number at most once.

For part 2:

Do merge sort on the array, We spilt the array in half per time with first half on left and second half on right. Until there is only 1 element in the subset. The array on the left will be set B and array on the right will be set C and use the algorithm in part 1. Also we will build an array with size (|C|+|B|). Since each time we compair, we can added to the bigger array sorted in the same time. The total running time will be O(nlogn), the same as mergesort.

CS/ECE 374, Fall 2018	Gradescope name:
Midterm 2: Problem 6	Xinrui Ying

Let $G = (V, E)$ be a directed graph with non-negative edge lengths $\ell(e), e \in E$ that represents a road network. Alice is invited to a party at her friend Bob's house and she wants to buy dessert at a grocery store on the drive to his house. Just as she is about to leave she realizes that she does not have sufficient gas in her car to drive all the way to Bob's house. Her car can go at most a distance of R before the gas runs out. Describe an efficient algorithm to help Alice accomplish the task of reaching Bob's house with as little travel as feasible; she needs to fill gas and buy dessert. Assume Alice's house is at node s and Bob's house is at node t and that the grocery shops are given by a set $X \subset V$ and the gas stations by a set $Y \subset V$. Assume that X, Y are disjoint sets. Also assume, for simplicity, that once Alice fills gas she can travel an infinite distance. Note that Alice could buy dessert either before or after filling up gas, as long as she does not run out of fuel on the way to the gas station. Express your running time as a function of n, the number of nodes, and m, the number of edges. Faster algorithms will earn more points but incorrect solutions earn very few if at all.

Do a dijkstra algorithm on every vertex of G. Then we want to consider the two following cases: 1. go to gas then to grocery.
2. go to grocery then to gas.
However, $dis(s, x) \le R$ $x \in X$ for both case.

$$\min \begin{cases} \min\{dis(s,x) + dis(x,y) + dis(y,t)\} & \forall x \in X, \forall y \in Y, \\ \min\{dis(s,y) + dis(y,x) + dis(x,t)\} & x \in X, y \in Y, dis(s,y) + dis(y,x) \le R. \end{cases}$$

The running time will be $O(nm + n^2 \log n)$ for dijkstra algorithm and $O(n^2)$ for the minimum computation since there will be no more than n^2 of pair (x, y) in all cases $x \in X, y \in Y$.
So running time $O(nm + n^2 \log n)$

This page for extra work.

This page for extra work.

This page for extra work.

This page for extra work.

This page for extra work.