# INTRO TO ALGORITHMS ASSIGNMENT 2 GROUP TASK

PEYTON GRATTINO & ALEX KUNZLER

## Contents

## 1. Graphs

### 1a) What is the cost of the minimum spanning tree?

The cost of the minimum spanning tree is 19.

### 1b) How many minimum spanning trees does it have?

There are two ways you can achieve this by swapping the edge E-B for B-F.

### 1c) Tour of minimal spanning tree.

| Node Start | Node End | Weight |
|---|---|---|
| A | E | 1 |
| E | F | 1 |
| B | F | 2 |
| B | E | 2 |
| F | G | 3 |
| G | H | 3 |
| C | G | 4 |
| B | C | 5 |
| C | F | 5 |
| D | G | 5 |
| A | B | 6 |
| C | D | 6 |
| D | H | 7 |

They would be added in the following order

| Node | Connection | Connection | Connection | Connection | Added |
|---|---|---|---|---|---|
| | 1-weight | 2-weight | 3-weight | 4-weight | Edge |
| A | E-1 | B-6 | | | A-E-1 |
| B | A-6 | E-2 | F-2 | C-5 | B-E \|\| F-2 |
| C | B-5 | F-5 | G-4 | D-6 | C-G-4 |
| D | C-6 | H-7 | G-5 | | D-G-5 |
| E | A-1 | B-2 | F-1 | | E-F-1 |
| F | E-1 | B -2 | C-5 | G-3 | F-G-3 |
| G | F-3 | C-4 | D-5 | H-3 | G-H-3 |

**2a) Prim's Sort.**

| Node Start | Node End | Weight |
|---|---|---|
| A | B | 1 |
| B | C | 2 |
| C | D | 3 |
| A | E | 4 |
| E | F | 5 |
| F | G | 1 |
| G | H | 1 |

A, B, C, D, E, F, G, H
A-H
1, 2, 3, 4, 5, 1, 1

**2b) Kruskal's Algorithm.**

| Node Start | Node End | Weight |
|---|---|---|
| A | B | 1 |
| B | C | 2 |
| C | G | 2 |
| F | G | 1 |
| G | H | 1 |
| G | D | 1 |
| A | E | 4 |

**3.**
Start at Node (1 || A) = x
For every edge branching from x, create * to go to that edge to the new node.
For every new node create a pointer to go down that edge.
When two pointers meet, delete one of the paths the pointer took.

If no pointers cross then there is no edge you can remove from G without leaving it unconnected.

## 2. Cost / Complexity / Big O

**Equation A.**
This is a divide and conquer algorithm, it has a running time of, **O(nLog(n))**

**Equation B.**
This is a recursive algorithm, and it has a run time of, **O(n)**

**Equation C.**
This is a 3-way merge sort, it has a running time of, **O(nLog(n))**

I would pick the divide and conquer, while it has a larger set-up that takes more memory it can sort the pieces quicker.

## 3. Algorithmic Efficiency

A linked sequence algorithm, you take the number you're looking for and find the spot in the array it should be, if it's not there then take the number you did find and go to the place in the array repeat until x is found. If you come back across a number you originally found, exit that loop and pick a different number, but the chance of this happening is very small.

## 4. Quicksort

**i.**

**ii.**
Since point p is the pivot if i = j and they equal p that mean they have both reached the pivot point which was declared as the center and would be the point that everything would be sorted around. Not only that but it does also sort the middle point so there is no point that is missed.

**iii.**
The reason why j cannot point to an element more than one position to the left of the one pointed at by i is because if it did then they would be overlapping or they would be missing elements in the sort and that would cause for an incorrect sort.

**iv.**
The example given in the book is a great example, of why quick sort is not stable. "... if an input list contains two equal elements in positions i and j where i < j, then in the sorted list they have to be in positions i' and j', respectively, such that i' < j'." (Levitin 19,20)[1]. Levitin also gives a great example and that is that if we were to sort students with their GPA's. If we have multiple students with the same GPA it should sort by name next but quick sort won't do that.

**v.**

*1. All elements are equal.*
This would be a worst case scenario and this is because it will still cycle through all values in the array, and attempt to sort it.

*2. Strictly decreasing.*
This would also be a worst case scenario because quicksort would leave one whole subarray empty after the split at the pivot point.

[1]Levitin, Anany. Introduction to the Design and Analysis of Algorithms. 3rd ed., Pearson, 2012