Exercise 1-1

20/25

asymptotic?

$5^{5^n} > 4^{4^n} > 4^{n^4}$

$4 \lg\lg n < 4 \lg n < (\lg n)^{5 \lg n} < n^{\lg n} < n^{n^{\frac{1}{5}}} < 5^n < 5^{5^n} < (\frac{n}{4})^{\frac{n}{4}} < n^{\frac{n}{4}} < 4^{n^4} < 4^{4^n} < 5^{5^n}$

Exercise 1-2

20,0/25 (7/10)

a) $a=4, b=4, c=1$ by masters theorem
$T(n) = O(n \log_2 n)$ 3rd case of masters

b) $a=4, b=5, c=5$ by master thm.,

$T(n) = O(n)$

c) $a=5, b=4, c=1$

$T(n) = O(n^{\log_4^5})$ 1st. of masters

d) $a=25, b=5, c=2$ 2nd case of masters thm.
$O(n^2 \cdot \log n)$

e) $a=4, b=5, c=\sim$
$\log_5^4 < \log_2 n$ (1) $T(n) = O(n^{\log_5^4})$

$c = \log(\log n)$

f) $a=4, b=5$ since c is too small, $\Rightarrow O(n^{\log_5^4})$

g) $T(n) = 4T(\sqrt{n}) + \lg^5 n$

h) $T(n) = 4T(\sqrt{n}) + \lg^2 n$

i) $T(n) = T(\sqrt{n}) + 5$

we side $\log n$ for n $\Rightarrow T((\log n) = T(\frac{\log n}{2}) + 5$ change var

$\therefore \log n$ steps with the tree, $\Rightarrow \log \log n$

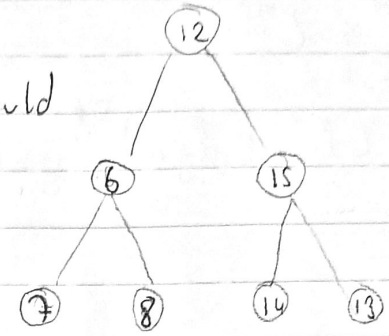
j) $\frac{n}{2}$ $\frac{2n}{5}$ $\frac{n}{10}$ } n

since everytime $\frac{n}{2}$ is biggest and takes most time to remove we take logn as height. $T(n) = O(n \log n)$

Problem 1-1

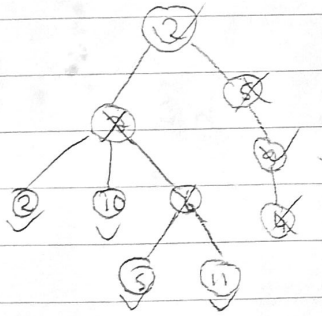
5/5

a) If our tree was like this with the given algorithm we would choose 15 and discard 13, 14, 12 which is not ideal and end in profit = 23 however, if we choose 13, 14, 12, 8 our $\Sigma p_v = 47$ which is better.



8/9

b) My algorithm is depth first. It will traverse the tree to reach bottom vertex, when it reaches bottom, it will go to parent node and look for other childrens. Algorithm will compare which is more profitable sum of children or parent nodes. If children are more profitable and these children are not directly conected then they will be added to V. If parent has one child they will be compared. After some node is decided all the nodes that are directly conected to it will be removed from tree. Example tree is below.

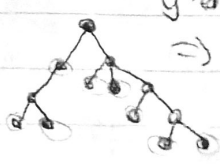


First it will compare 5+11 and 6
5+11 is chosen, 6 is removed
2+10 > 7 2 and 10 chosen 7 removed
9 > 4 so 9 chosen 4 removed 5 removed
2 chosen
sol = 39 This sol. works for every tree.

5/6

↳ since we iterate n times in dfs it is automatically O(n).

c) Here, best idea, would be go to deepest nodes possible and take them as solution until no nodes are left. Since when we take a child node as solution and put it into V we also remove the parent after we are done with all child nodes we can recursively call the algorithm until no nodes left. Since always number of children ≥ parent(1) this would always work. yields in ⇒ 8

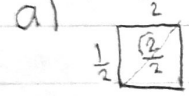


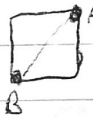
** Another solution could be sort nodes by number of edges on them and take ones with least and more.

d) Searching all nodes, if a nodes profit is higher than sum of nodes it is directly connected add it to V. Else;
Go to a node with 1 edge, if it's neighbours profit is less than the nodes connected to that neighbour, take the node to V and if possible take all other nodes except neighbour starting from most profitable. After recursively doing this, pick all that is left.
Complexity: Since we check all nodes and do calculations, generally this can be considered O(n) but, at worst case where each node is connected everywhere it can be $O(n^2)$.

2/5

Problem 1-2.

a) 

With a square with dimensions 0,5 x 0,5 since if two requests are within distance of 1 and because two points on a single square are at most diagonally distant from each other if we prove 

dist(A,B)<1

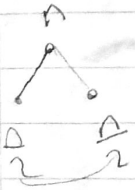
3/3

we prove for each given pair in or on a square.
Since $\frac{\sqrt{2}}{2} < 1$ then we must reject any two requests if they are in some circle.

b) 1) First divide the entry points from middle
2) Keep dividing until in a division there is 1 or 2 points. If we have two points in a square since we proved with a) they are closer than 1 and reject and return them. If they are not in some circle then calculate distance between.

14/14

4) This algorithm is very much like the divide and conquer algorithm we saw on the lecture. Say, all points are on the convex  The algorithm will divide as on the example and since algorithm works for 1, 2 solutions it can also work with P solutions.



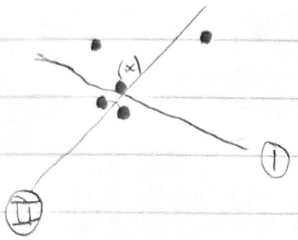
*) Complexity can be calculated with;
since for n many point requests, n-1 division lines or drawn this can be changed to n or n+1 etc but it is O(n) in any case. From the left we have n many "works" in each step and log n steps so
$T(n) = n \log(n)$

2/4

7/8

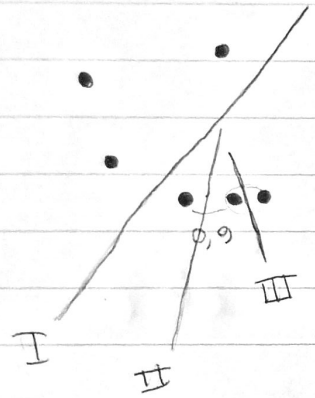
c) After finding two points within distance of 1 just like the solution in b), go one iteration up and execute the merging operation which is look for another point of the opposite side of line and if it is within the 1 unit distance to either of the points we found then return these 3 points otherwise move to other close couple points. Illustration is below

Say, we have canvas like below.



First draw line ①

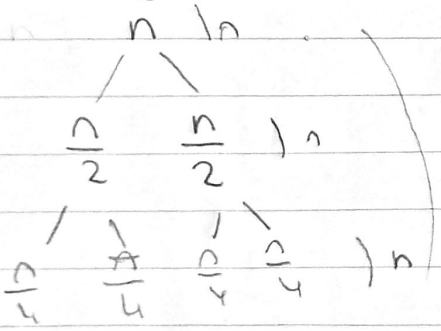
After second line algorithm finds two points then it removes line ② checks if (x) is within 1 distance of found points.



Now I is drown, II, III are drawn. Near the III two points that are close are found.
Line III is removed and algorithm looks for points of opposite side of II. It finds that one is in 1 dist (0,9) but other is not so they are not a match.

Running time:

Since algorithm in b) is n log n because if we draw a tree,



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

also because in each found pair with n log n we only do set number of operations which are finding distance between 2 points we have $O(n \log n) + O(2)$ which is $O(n \log n)$.

84/100

Complexity of b) in problem 1-1

Since this algorithm traverls each node and dfs is O(n) and also only constant number of operations are done in each node that is comparing sum of childs to parent then the whole algorithm is also O(n).