Question 1

9 pts

Which of the following statements are true regarding the standard version of the stable matching problem given *n* men *n* women. Select all that apply.

- [ ] Since the standard propose-and-reject algorithm is female-pessimal, in any matching assigned by this algorithm, each woman will be matched with the last person on her respective preference list.
- [ ] The standard propose-and-reject algorithm seen in lecture may find a female-optimal solution.
- [ ] Given any instance of *n* men and *n* women with their corresponding preference lists, there may exist more than one stable matching that is male-optimal.
- [ ] The main while-loop of the standard propose-and-reject algorithm seen in lecture may run for exactly n iterations and then terminate.
- [ ] An unmatched pair *(m,w)* is unstable with respect to a matching *M* if man *m* and woman *w* prefer each other to their current partners assigned by *M*

Question 2

9 pts

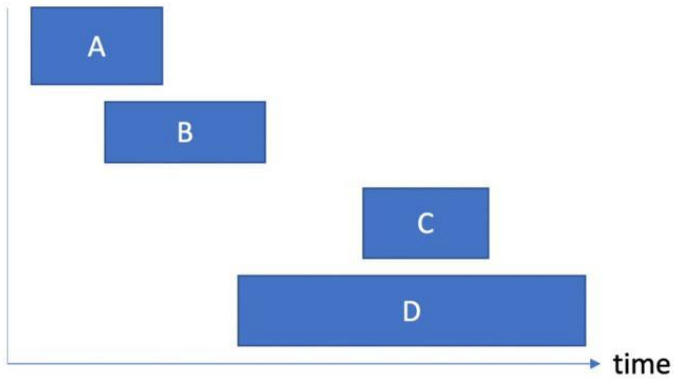
My friend has a factory with two identical machines that each may process the same types of job. As such, my friend can schedule as many as two jobs during any given time interval. In addition, he always has a selection of jobs he may choose to run, but each of these jobs has a corresponding fixed time interval in which it may run, that is, each job has a fixed start time and a fixed finish time. He tells me he knows a greedy algorithm that will produce an optimal schedule for this **interval scheduling** variant (i.e. at most two jobs may be scheduled at any point in time). If J is the set of all fixed job intervals that may be processed, we run the following algorithm:

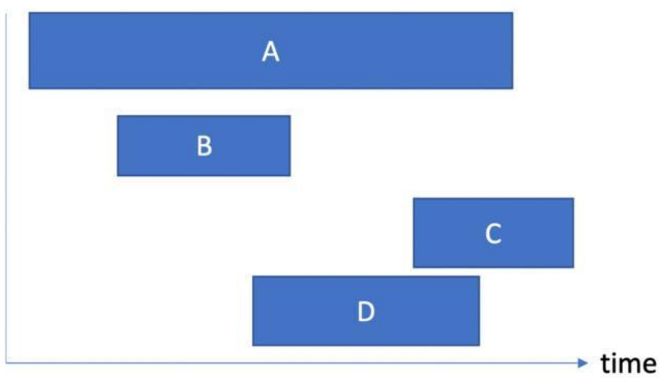
Algorithm 1 DOUBLE GREEDY INTERVAL SCHEDULING

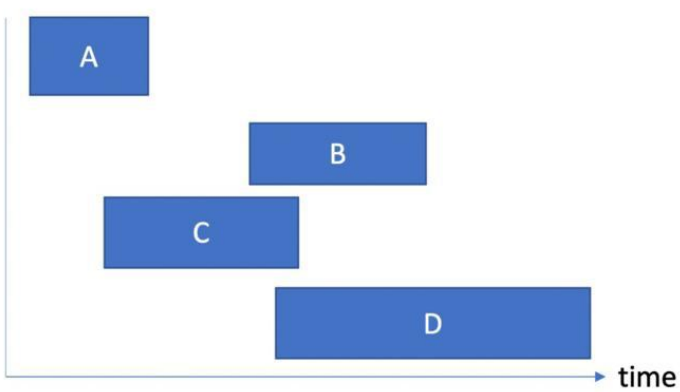
Input: Set J
Sort the elements of J in non-decreasing order of finish time
$A_1 \leftarrow \emptyset$
for $j = 1, 2, \ldots, |J|$ **do**
  if job j does not overlap at any point with any job in A_1 **then**
    $A_1 \leftarrow A_1 \cup \{j\}$
Set $J \leftarrow J \setminus A_1$ (i.e. remove the contents of A_1 from J)
$A_2 \leftarrow \emptyset$
for $j = 1, 2, \ldots, |J|$ **do**
  if job j does not overlap at any point with any job in A_2 **then**
    $A_2 \leftarrow A_2 \cup \{j\}$
Return $A_1 \cup A_2$

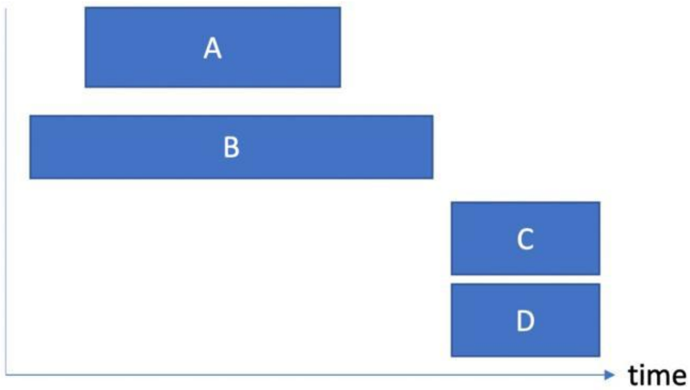
My friend says DOUBLE GREEDY always produces an optimal schedule for the variant, but he is, in fact, wrong. From the selection of job interval sets below, choose the selection that functions as a simple counterexample that shows my friend is incorrect.

My friend says DOUBLE GREEDY always produces an optimal schedule for the variant, but he is, in fact, wrong. From the selection of job interval sets below, choose the selection that functions as a simple counterexample that shows my friend is incorrect.









Question 3

9 pts

Give the reason(s) that for any valid flow network $G = (V, E)$, a maximum flow always exists. **Select all that apply.**

- [x] The edge capacities of G are non-negative
- [x] There is exactly one source and one sink node
- [ ] The flow f is defined as $f(e) = 0, \forall e \in E$ is a candidate flow.
- [ ] One may always completely saturate every edge with flow to obtain a valid maximum flow

Question 4

9 pts

Below is a list of statements regarding NP-completeness and reducibility. **Select all statements that are true.**

- [x] The VERTEX-COVER problem can be reduced in polynomial time to the SET-COVER problem: The implication of this reduction is that if we are ever able to solve the VERTEX-COVER problem in polynomial time, we would also be able to solve the SET-COVER problem in polynomial time.
- [x] Suppose $X \le_P Y$. If the size of the instance of X is n, then we can transform an instance of X into an instance of Y in time $O\left(n^k\right)$ for some constant k.
- [ ] If problem X is NP-hard, then it follows that X is necessarily not in NP.
- [x] SET-COVER$\le_P$VERTEX-COVER.
- [ ] Given decision problems X and Y, we define a new type of reduction $X \le_E Y$, where $\le_E$ follows the same definition as seen in lecture for the polynomial time reduction $\le_P$, but for the fact that we allow time exponential in the size of the instance of X for this instance to be transformed into an instance of Y. Then given $X \le_E Y$, if we were ever able to solve Y in polynomial time, it would imply we can solve X in polynomial time.

The CLIQUE problem is in NP since one can verify that a certificate, i.e. a subset of vertices, for an instance of this problem is valid by comparing the vertices in the given certificate with the corresponding subset of vertices of the instance and checking that there exists an edge between each of these vertices. This can be done in polynomial time.

We show INDEPENDENT-SET $\leq_P$ CLIQUE. From an instance G of CLIQUE, we create a complement graph G' as described in the prompt hint. We then argue G' has a(n) clique of size k iff has G a independent set of size k.

($\Rightarrow$) Suppose G' has a(n) clique of size k. Then since the presence of an edge in G' corresponds with the absense of an edge in G, and since a clique is a set of vertices such that every pair shares an edge, there exists a set of vertices in G of size k such that no two vertices share an edge, i.e. a(n) independent set.

Question 6 9 pts

If $X \leq_P Y$ and Y is NP-complete, what can we then conclude about X?

Select all that apply.

- [x] None of the other statements is a valid conclusion about X.
- [ ] X is NP-hard.
- [ ] X is in NP and not in P.
- [ ] X is NP-complete
- [ ] X is in P

◂ Previous Next ▸

Question 7

9 pts

Consider the VERTEX-COVER problem seen in lecture, where given a graph $G(V, E)$, one would like to find the minimum cardinality subset V' of V such that V' is a vertex cover of G. It has been shown this problem is NP-hard in lecture.

Suppose I have invented an approximation algorithm that finds a suboptimal solution to the VERTEX-COVER problem and I have somehow proven that this algorithm is a ρ-approximation of the optimal solution. What can one then conclude about the value of ρ?

Select all that apply.

- [ ] Nothing meaningful can be said about ρ with the information provided.
- [ ] $\rho = 1$
- [x] $0 < \rho < 1$
- [ ] $\rho > 1$

◂ Previous   Next ▸

Question 8

9 pts

Below is a list of statements regarding the PTAS for the KNAPSACK problem, which runs in $O\left(\frac{n^3}{\epsilon}\right)$ time, where n is the number of items in the knapsack, and ϵ is a parameter used in the PTAS algorithm . **Select all statements that are true.**

- [ ] The dynamic programming framework used to design the KNAPSACK PTAS seen in lecture considered subproblems of the form $OPT(i, v)$, which was defined as being the minimum weight subset of items $1, \ldots, i$ that yields a combined value of **at most** v.
- [x] The solution for the PTAS for KNAPSACK is **at least** $\frac{1}{1+\epsilon}$ times the optimal solution for any given instance.
- [ ] As ϵ decreases, the time required until the PTAS terminates increases.
- [x] A valid value for ϵ in the PTAS is 2^{-n}, since ϵ can be arbitrarily small.

◂ Previous   Next ▸

1. The PTAS for KNAPSACK can be described as follows: The algorithm reduces the values of each item in the knapsack down using a variable scaling factor such that the scaled down maximum value of any item is polynomial in n. It then returns the subset of items that corresponds to the solution of the instance containing the scaled down values.
2. A valid value for ϵ in the PTAS is 2^{-n}, since ϵ can be arbitrarily small.
3. The solution for the PTAS for KNAPSACK is **at least** $\frac{1}{1+\epsilon}$ times the optimal solution for any given instance.
4. The dynamic programming framework used to design the KNAPSACK PTAS seen in lecture considered subproblems of the form $OPT(i, v)$, which was defined as being the minimum weight subset of items $1, \ldots, i$ that yields a combined value of **at most** v.
5. As ϵ decreases, the time required until the PTAS terminates increases.

Part A (16 pts):

In lecture we saw the Knapsack: Dynamic Programming II (KNAPSACK II) framework, which is an optimization problem where we are given a set S of n items, each with an associated integral weight and a profit w_i, v_i, respectively. With this, we seek to find a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} w_i$ is minimized and $\sum_{s_i \in S'} v_i = V$ for an arbitrary integer V. (On a side note, this formulation was related to the original Knapsack framework (KNAPSACK I) from module 5 by setting $V = V^*$ where V^* is the largest V such that $\sum_{s_i \in S'} w_i \leq W$ for an arbitrary integer W). One could formulate the decision version of KNAPSACK II by including an additional integer W, and asking whether there exists a subset $S' \subseteq S$ such that $\sum_{s_i \in S'} v_i = V$ and $\sum_{s_i \in S'} w_i \leq W$.

As seen in lecture (and by many of you in midterm exam 2), the SUBSET-SET problem is a seemingly related decision problem where we are given a set Z of k integers and an arbitrary integer X. For these we ask whether there exists a subset $Z' \subseteq Z$ such that $\sum_{z_i \in Z'} z_i = X$. Assume that SUBSET-SUM is NP-Complete, and use this to show that the decision version of KNAPSACK II is NP-Complete. Of the three reduction strategies seen in lecture, which one did you use? Explain.

Part B (6 pts):

Is the optimization version of KNAPSACK II NP-Complete? Either prove this is the case, or argue why the problem may not be NP-Complete. You may cite your answer to part a in your response if necessary.

Part C (6 pts):

Give a description of a $\frac{1}{2}$-approximation algorithm for KNAPSACK I (from module 5) . Justify correctness and runtime. You may directly cite lecture materials as needed without reproducing them.

~~Subset problem~~

Knapsack II problem:

$OPT(i,v)$ = min weight subset of item $1 \ldots, i$ that yields value exactly v.

using OPT method, we can define OPT function as

$$OPT(i,v) = \begin{cases} 0 & \text{if } v=0 \\ \infty & \text{if } i=0, v>0 \\ OPT(i-1,v) & \text{if } v_i > v \\ \min\{OPT(i-1,v),\ w_i + OPT(i-1, v-v_i)\} & \text{otherwise} \end{cases}$$

writing above problem as decision problem, we can define Knapsack problem ~~~~ as: Given a finite set S of n items, each with an associated integral weight and profit w_i, v_i resp. with this, we seek to find a subset $S' \subseteq S$ such that weight of sub is minimalized, and necessary target value V is obtained.

$$\sum_{s_i \in S'} w_i \text{ is minimalize}$$

$$\sum_{s_i \in S'} v_i = V$$

for $s' \subseteq S$ $\quad \sum_{s_i \in S'} v_i = V$ & $\sum_{s_i \in S'} w_i \leq W$

OR

Given a finite set S, nonnegative weights w_i, non negative value v_i, a ~~weight limit W, and~~ a target value V can be obtained with minimalising weights, is there a subset $S' \subseteq S$ such that

Subset-Sum :- Given a finite set Z of k integers, and arbitary integer X, is there a subset $Z' \subseteq Z$ such that whose elements sum to exactly X.

$$\text{i.e} \sum_{z_i \in Z'} z_i = X.$$

Based on the question, let us assume subset-set is NP-Complete.

To prove knapsack is NP-complete, I would like to claim & prove subset-sum $\leq_p$ knapsack using spc to general reduce an instance of subset prob to an instance of knapsack prob

Given instance of subset $(z_1, \dots z_k, X)$ create knapsack instance such that

$$\left\{ \begin{array}{l} v_i = w_i = z_i \\ V = W = \cancel{X} \end{array} \right. \qquad \begin{array}{l} \sum_{i \in S'} z_i \leq X \\ \sum_{i \in S'} z_i \geq X \end{array}$$

OPT(i, v) = min weight subset of items $i \dots i$ that yields value 'V'.

Yes/No to the new problem (instance of knapsack corresponds to the answer for the original problem.

$$\left\{ \begin{array}{l} \sum_{i \in S'} v_i \leq V \Leftrightarrow \sum_{i \in S'} z_i \leq \cancel{X} \\ \sum_{i \in S} w_i \leq W \Leftrightarrow \sum_{i \in S'} z_i \leq X \end{array} \right. \Rightarrow \sum_{i \in S'} z_i = X$$

We have proved that instance of Knapsack can generate solution to subset prove.

The reason we used above reduction strategy is because ~~we are aware that subset sum is already NP-complete prob~~
~~Hence, can poly-reduce~~ Knapsack is considered as special case meeting with respect value V with min weigh W to general case of subset problem where sum of values in subset to be X.

Knapsack is NP-Complete

as if $X \leq_p Y$ and X is NP-Com

then Y cannot be determined to be solved in polynomial time

which makes Y also NP-complete [principle of polynomial time reduction]

Subset $\leq_p$ Knapsack

Hence Knapsack is NP-Complete

Part B

Part C: 1/2 approximation indicates Approximate $1/1+\epsilon$ to be $1/2$

W_{max} = largest weight in original instance

$O(n W_{max}) = O(n^3/\epsilon)$

Appr. algo $= \frac{1}{1+\epsilon}$ optimisation Algo

$\frac{1}{1+\epsilon} = 1/2$

$\epsilon = 1$

If 'S' is solution found ... any ~~other~~ other feasible solution then

$$(1+\epsilon)\sum_{i \in S} v_i \geq \sum_{i \in S^*} v_i$$

$$\rho = \frac{1}{1+\epsilon}$$

W_{max} = Largest value
ϵ = precision para
θ = scaling factor
$\epsilon V_{max}/n$

Run time $\rightarrow O\left(n^3/\epsilon\right)$

Part B:

Optimising solution is Dynamic Solution

For knapsack II definition from ~~...~~ Part A:-

V^* = optimal value = maximum V such that $OPT(n,V) \leq W$

* Not Polynomial in Input Size

App = 2OPT

= $\frac{1}{2}$ OPT

Appr = 2OPT

Appr. = $\frac{1}{2}$ OPT

Part: C :-

Knapsack problem has PTAS following
$(1+\epsilon)$-approximation alg for any constant $\epsilon > 0$.

For the question :- $1/2$ approximation

ϵ has to be $-1/2$ which contradicts the definition.

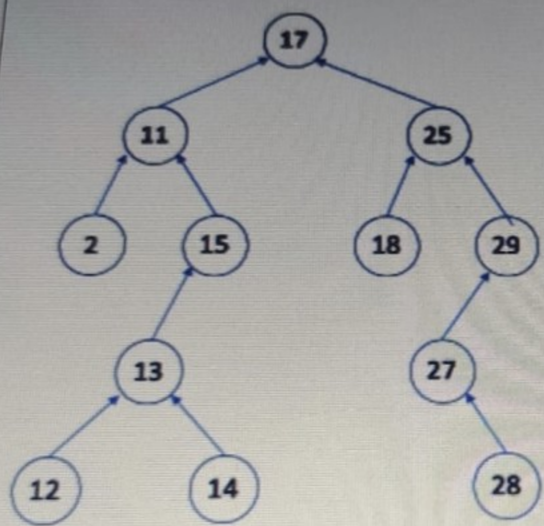
Hence it concludes that there is no optimised solution only for $1/2$ approximation algorithm for knapsack.

Question 10

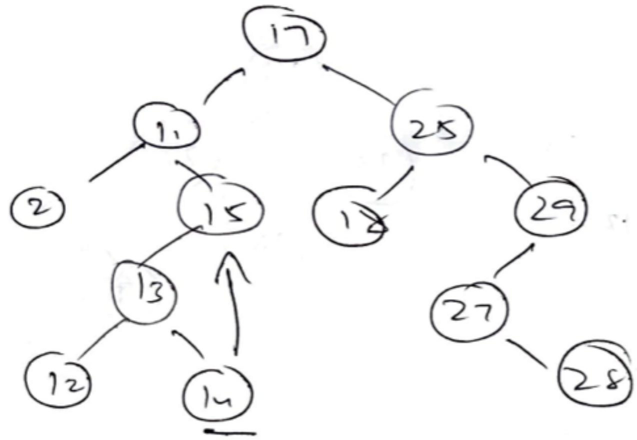
0 pts

The following question is an optional bonus question worth 5 points to your total exam score.

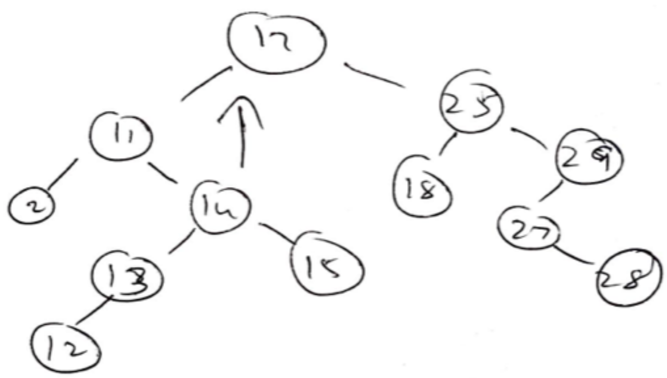
Given the following splay tree T, obtain the resulting configuration after the operation **SPLAY-DELETE(T,14)** is applied. Recall that the SPLAY-DELETE operation splays the *predecessor* of the deleted node. **For full marks, show the configuration of T after each double rotation (and possibly one final single rotation).**



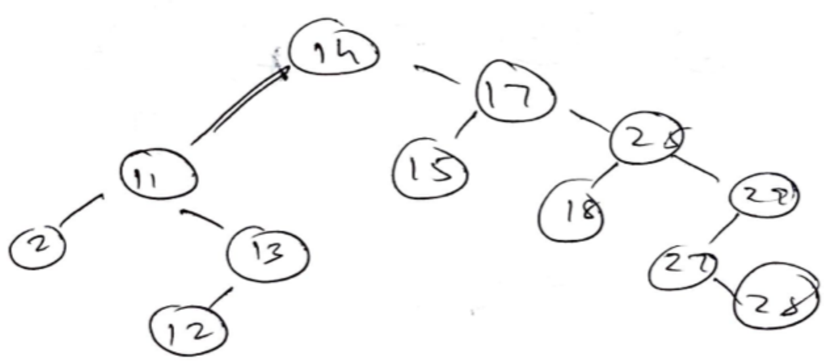
10



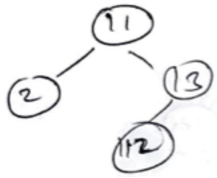
Splay-14:-

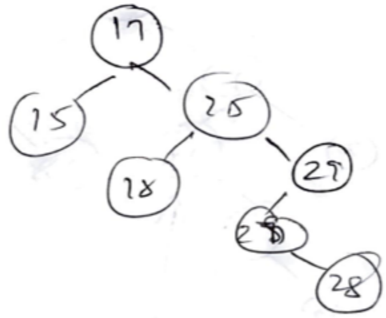


⇓ Zig-zag

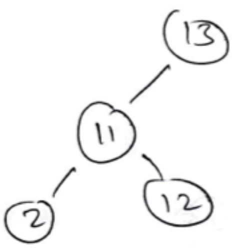


⇒ Delete 14 and splay 13 from left tree

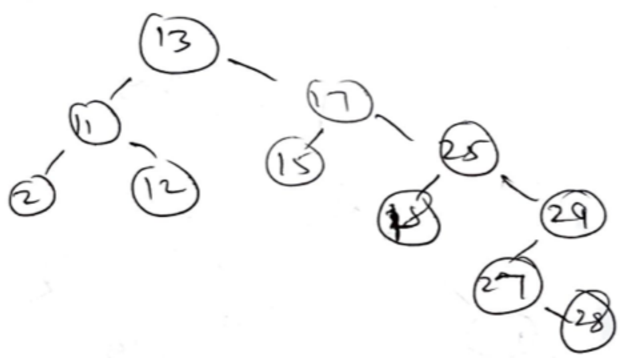




splay - 13



attach right side



x

Question 11

0 pts

The following question is an optional bonus question worth 5 points to your total exam score.

The run-time of the Knapsack problem is $\Theta(nW)$. Assume the input is represented in binary. Is this polynomial run-time? Why/why not? **Select all that apply.**

- [ ] Yes, because W is an integer
- [ ] Yes, because $nW \leq n^2$, which is polynomial.
- [x] No, it can only run in polynomial time if W is some polynomial function of n
- [x] No, because W is represented in binary, and the value of W is exponential in $\log(W)$

Decision pblms - Yes/No | $P \subseteq NP$

NP - decision pblm - can be verified in poly time

$|P| \leq NP$ $P \neq NP$ (mostly)

$X \leq_p Y \Rightarrow I_X$ can be reduced to I_Y in poly time

Reduction strategies:

1) Simple equivalence: Ind set $\equiv_p$ Vertex Cover
2) Special case to Gen case: Vertex Cover $\leq_p$ Set cover
3) Reduction by encoding gadgets: 3 SAT $\leq_p$ Ind set

Interval Sched $\leq_p$ Ind Set ($\because$ Ind set is NP complete)

NP complete:

1. Y in NP
2. Find known NPC X
3. Prove $X \leq_p Y$

ρ Approx Algorithms: $0 < \rho < 1$

NP complete - SAT / CIRCUIT SAT

Approx Algo: $(1+\epsilon)$ optimal soln.

Subset Sum $\leq_p$ Knapsack

Knapsack is NPC

FPTAS: $\theta = \frac{\epsilon v_{max}}{n}$; θ - scaling factor, ϵ - Precision factor

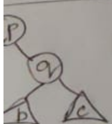
$\bar{v}_i$ (Rounded) $= \left\lceil \frac{v_i}{\theta} \right\rceil \theta$

$\hat{v}_i$ (Approx) $= \left\lceil \frac{v_i}{\theta} \right\rceil$ | Optimal soln to prob with $\bar{v}$ or $\hat{v}$ are =

$E[X] = \sum_{i=1}^{n} (E[X_i]) = \sum Pr(Y_i = 1)$

$w(MST) \leq w(TSP) \leq 2w(MST)$

$s(\sqrt{n}) + s(n - \sqrt{n}) = O(n^{3/2})$, $O(n^{3/2}) > O(n \log n)$



$r'(q) - r(q)$

k of v:

$) = \log(|V|)$

Am cost of splaying a node is atmost:

$3 \log n + 1 = O(\log n)$

Am cost of: i) Single Rot: $1 + 3r'(v) - 3r(v)$

ii) Double rot: $3r'(v) - 3r(v)$