CS 5633 Analysis of Algorithms – Spring 14

Exam 2

NAME: Tanvir Irfan Chowdhury

- This exam is closed-book and closed-notes, and electronic devices such as calculators or computers are not allowed. You are allowed to use a cheat sheet (half a single-sided letter paper).
- Please try to write legibly – if I cannot read it you may not get credit.
- **Do not waste time** – if you cannot solve a question immediately, skip it and return to it later.

1) Red-Black Trees		15
2) B-Trees		15
3) Greedy Algorithms		25
4) Dynamic Programming		25
5) Amortized Analysis		20
	87	100

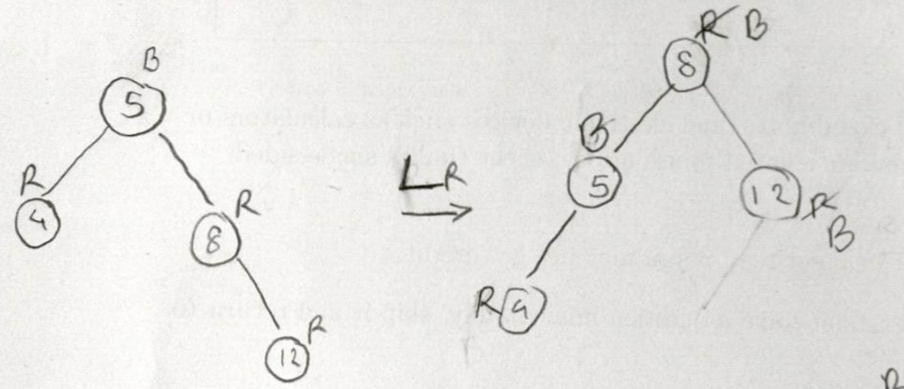
8

1 Red-Black Trees (15 Points)

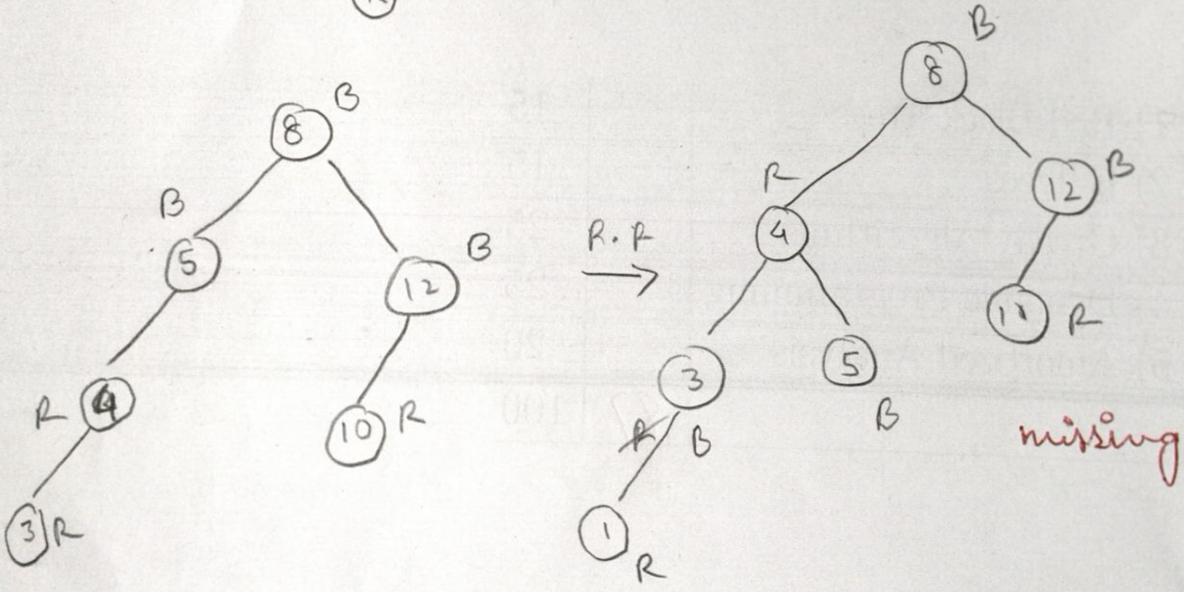
Consider making the following sequence of inserts into an initially-empty red-black tree.

$$5, 8, 4, 12, 10, 3, 1$$

Show the tree before and after each rotation or recoloring (i.e. you do not need to draw it after each insertion if the insertion does not force a rotation or recoloring).



-7



missing cases!!

t = 3 , AME 5 , AL = 2

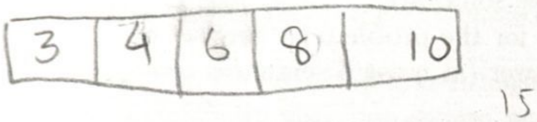
13

2 B-Trees (15 Points)

Consider making the following sequence of inserts into an initially-empty B-tree with $t = 3$.

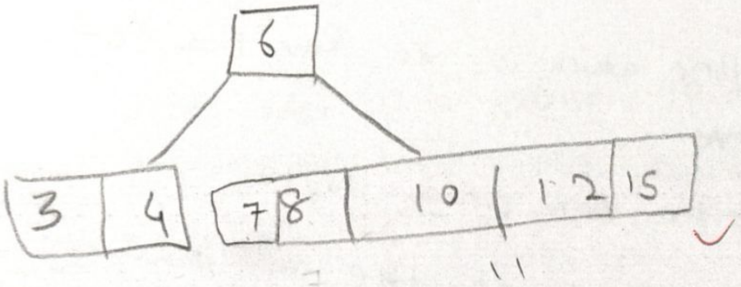
$$3, 6, 10, 4, 8, 12, 15, 7, 11$$

Show the tree before and after each split (i.e. you do not need to draw it after each insertion if the insertion does not force a split).

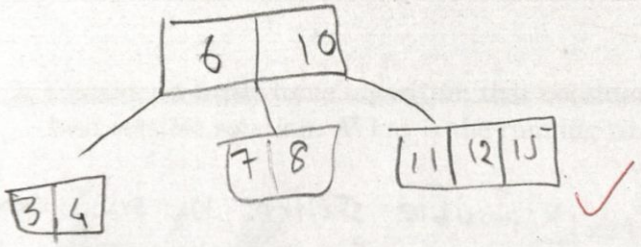


✓ show the split after insert 12

−2



✓



✓

23

3 Greedy Algorithms (25 Points)

Suppose you are going to go on a hike along the 2,000+ mile Appalachian Trail. Suppose you can hike at most 20 miles per day, and therefore will have to spend many nights camping along the trail. The trail has n shelters $s_1, s_2, \ldots, s_n$ along the way, and you are required to spend each night at a shelter. For each i from 1 to $n-1$, let d_i denote the distance between shelter s_i and shelter s_{i+1}. Your goal is to compute a sequence of shelters at which you will stay to minimize the number of nights spent hiking the trail. Note that since you can hike at most 20 miles per day, it is required that your chosen shelters be at most 20 miles apart. Assume that each d_i is at most 20 (otherwise there is no feasible solution).
Give a greedy algorithm that will compute an optimal solution for the problem for any set of input shelters. Argue why your algorithm returns a correct answer (at most 5 sentences or so should suffice for this argument).

I am at the very beginning, distance covered ~~=~~ $\boxed{d = 0}$

Algorithm Steps -

1. Choose the farthest shelter which is no more than $(d+20)$, spend the night there

2. $d = d +$ distance of shelton from previous position.

3. repeat ① & ② until reach at the end point, s_n.

Correctness:

- I started taking rest at the farest possible shelter. No sol^n can do better than my sol^n according to the given constrains.
- I maximize the "distance covered" by choosing the next shelter which is "Just inside" my reach of "20 miles". No other sol^n can beat it too.
- So, my sol^n is optimal.

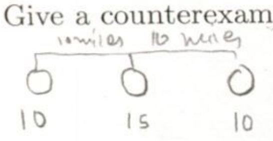
Right idea but substitution method needed for proofs −2

23

4 Dynamic Programming (25 Points)

This problem is similar to the greedy algorithms problem (you should read that question before reading this problem). We again want to hike the Appalachian Trail and can hike at most 20 miles per day. There again are n shelters $s_1, \ldots, s_n$, but now suppose there is a positive cost c_i that is charged to hikers for using shelter s_i. For simplicity, assume that "adjacent" shelters are 10 miles apart. The goal is to compute a sequence of shelters along the trail such that the sum of the costs of the chosen shelters is minimized. Note that in this situation, we are willing to spend additional nights on the trail if it means that the sum of the costs of our chosen shelters is minimized.

1. Consider the following greedy algorithm. Repeatedly choose the shelter with the minimum cost until our chosen set of shelters allows us to reach the end of the trail. Give a counterexample to show that this algorithm fails.



greedy algo will choose 10 + 10 but clearly the optimal sol^n is 15.

✓

2. Consider a brute force algorithm that considers every subset of shelters and returns the best feasible solution. What is the running time of this algorithm?

$\frac{'x'}{s_1}$ $\frac{'x'}{s_2}$ $\frac{'x'}{s_3}$ - . . . . $\frac{'x'}{s_n}$

for each subset we can have 0, 1 for 'x' in each positⁿ.

so, there is 2^n possibilities

running time will be 2^n [as we will check all possible combinations' and determine the smallest].

-2

$2^n \cdot n$

↑

time to check feasably.

3. Let $a[i]$ denote the optimal cost to hike to shelter s_i. Note that this does not necessarily mean that we must stay at shelter s_i. For example, if we stay at shelter s_{i-1} then we can walk to shelter s_i in an additional 10 miles and would not necessarily have to stay at shelter s_i.

Suppose we have 5 shelters with the following costs: $A = \{5, 10, 4, 2, 8\}$. What is $a[1], a[2], a[3], a[4]$, and $a[5]$?

~~10 20 30 40 50~~

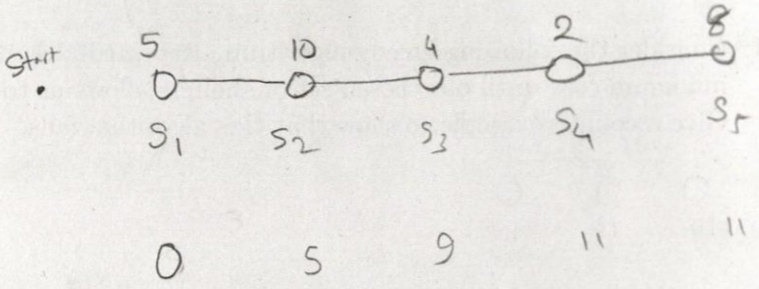
a[1] = ~~8~~ 0 // we can skip this to next shelter.

a[2] = 5

a[3] = 5+4 = 9

a[4] = 11

a[5] = 11



Ok

4. Give a recursive definition for $a[i]$. Do not forget the base case.

a[0] = 0 // at the beginning we didn't use any shelter.

$\boxed{B[0] = 0,\ B[1] = A[1] = 5;}$

if not stayed at (i-1) you should use (i-2)

$$a[i] = \begin{cases} \min\{a[i-1],\ a[i-2] + A[i]\} & \boxed{i > 2} \\ \min\{B[0] + A[2],\ B[1] + A[1]\} & \boxed{i = 2} \end{cases}$$

is stayed at (i-1)

~~a[i] → possible options are s_i is chosen to stay - in that case, if T[i-1~~

Ok

B[1-n] will track which site was used to stay in the "optimal sol^n".

20

5 Amortized Analysis (20 Points)

Recall the dynamic array data structure that was asked in class. If our data structure currently has size k, then we double the size to $2k$ once we receive operation $k+1$ and copy all of the elements from the old data structure to the new one. It is natural to view this doubling as quite wasteful when k is large. Suppose instead of always doubling the size of the array, we instead increase its size by 100 when needed (i.e. going from k to $k+100$ rather than $2k$). This question will consider the amortized cost of an operation in this data structure.

1. Perform an aggregate analysis to obtain a bound on the sum of the cost of n operations.

i	1	2	3	4	5	6	100	101	102	103 ...
Size	1	101	101	- - - - -			101	101	201	201 ---
c_i	1+0	1+1	1+0	- - - - - -			1+1	1+0	1+101	---

$$\sum_{i=1}^{k} c_i = n + [1 + 101 + 201 + \dots]$$

$$= n + [\overset{\frac{n}{100}\text{ times}}{1+1+\dots+1}] + 100[1+2+3+\dots\tfrac{n}{100}]$$

$$= n + \frac{n}{100} + 100 \cdot \frac{\frac{n}{100}\cdot(\frac{n}{100}+1)}{2}$$

$$\le n + n + 50 \cdot \frac{n^2}{100^2}$$ ✓

$$= 2n + \frac{n^2}{200}$$

$$= O(n^2)$$

2. Use the accounting method to determine the amortized cost of a single operation.

$$\sum c_i \le 2n + \frac{n^2}{100} \Rightarrow \text{per instruction cost} = \frac{n}{100} + 2$$ ✓

assume amortized cost for per instruction is $(100n + 1) → $\hat{c}_i$

$1 will be used to do its own work

$100n will be used for future increase.

Now Here is the bank balance —

$\boxed{n = 8}$

i	1	2	3	4	5	6	7	8
c_i	801	801	801	801	801	801	801	801
bank$_i$	800	1501	2301	... -				

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (100n+1)$$

$$= 100 \cdot \frac{n \cdot n+1}{2} + n$$

$$= 50n^2$$

$$\ge \sum c_i$$

we will never run out of money.