MIDTERM with SOLUTION

1. PLEASE READ INSTRUCTIONS CAREFULLY.
2. Answer all questions.
3. Use ONLY THE FRONT SIDE OF EACH PAGE for your answers.
4. Use the REVERSE SIDE of each page for scratch work.

GENERAL COMMENTS ABOUT YOUR ANSWERS:

- *Remember that grades for this course is curved (normalized) and not based on absolute points.*
- *Question 1 asked for "brief justifications" but many simply wrote down an answer and left it at that! Question 2 asked for intermediate trees after EACH insertion/deletion/rotation, but some simply skip many crucial intermediate steps. Please realize that you are not entitled to any points when you ignore such explicit instructions (but we have graded more generously than this).*
- *Each question asks for something specific: Question 1(iii) asked for a set, but most answers never even mention a set. Question 1(iv) asked for a solution a recurrence. You must give us what we asked for. Otherwise, you are not answering the question.*

1. **(10 points each part) Short Questions**
   You must give a brief justification (two or three sentences).
   (i) What upper and lower bound does the Master Theorem give for the following recurrence? $T(n) = 16T(n/2) + \frac{n^4}{\log\log n}$.
   (ii) Give the Θ-order of the following sum: $S(n) = \sum_{i=1}^{n} i^{2^i}$.
   (iii) You are given an algorithm A which, given two sequences $t = (u_1, u_2, \ldots, u_m)$ and $p = (v_1, v_2, \ldots, v_n)$, it will tell you whether p occurs as a subsequence in t. For instance $p = (a, a, b)$ occurs in $t = (a, a, a, b, a)$, but p does not occur in $t' = (a, b, a, b, a)$. Algorithm A takes time $O(m + n)$. Describe in no more than 2 sentences how you can test if two closed paths $p = (u_0, u_1, \ldots, u_k)$ and $q = (v_0, v_1, \ldots, v_\ell)$ are equivalent (i.e., represent the same cycle).

   SOLUTION:

   (i) $T(n) = O(n^4 \log n)$ since $T(n) \le T_1(n)$ where $T_1(n) = 16T_1(n/2) + n^4 = \Theta(n^4 \log n)$.

   Also, $T(n) = \Omega(n^4)$ since $T(n) \ge T_2(n)$ where $T_2(n) = 16T_2(n/2) + 1 = \Theta(n^4)$.

   (ii) This is an exponential type sum, and hence by our summation rule, $S(n) = \Theta(n^{2^n})$. To see that it is exponential type, we must show that $i^{2^i} \ge C(i-1)^{2^{i-1}}$. This is clearly true as $i^{2^i} = i^{2^{i-1}2} = (i^{2^i-1})^2 = C \cdot i2^{i-1}$ where $C = i^{2^{i-1}} \ge 4$ and $i \ge 2$.

   PITFALLS: Please do not confuse i^{2^i} (double exponential) with i^{2i} (single exponential).

   (iii) Here is the algorithm: Just check $\ell = k$ and use Algorithm A to detect that p is a subsequence of $t = (v_0, v_1, \ldots, v_\ell, v_1, v_2, \ldots, v_\ell)$.

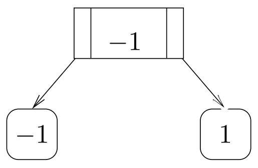
   PITFALLS: do not attempt to re-do the $O(n^2)$ algorithm in the homework! If you do not use algorithm A in your solution, you get NO credit.

2. **(10+10+20 points) AVL and (a, b)-trees**

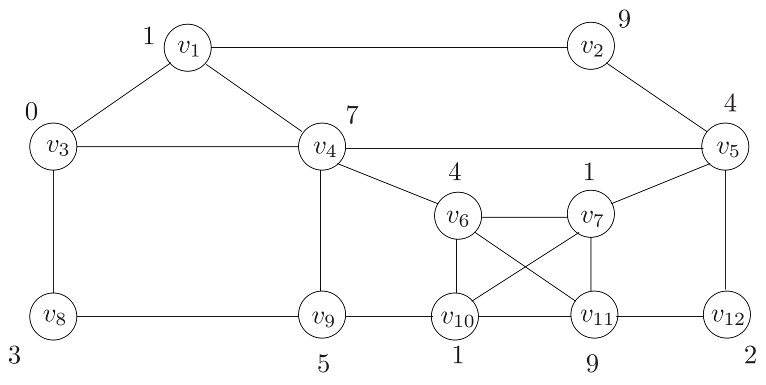
   We want to insert the following sequence of keys into an initially empty search tree:

$$1, -1, \frac{1}{2}, \frac{-1}{2}, \frac{1}{3}, \frac{-1}{3}, \ldots, \frac{1}{n}, \frac{-1}{n}.$$

(i) Let $n = 5$ (so you are inserting 10 keys). Draw the AVL tree at the *end* of each insertion. So we want to show us 10 trees for this question. You get one point per tree, and if any tree is wrong, then all subsequent trees are considered wrong. Do NOT show us your intermediate results, although you probably want to do this on the scratch pages. Please be careful, as it is easy to make mistakes.
(ii) Inserting the 10 keys of part (i) into a $(2, 3)$-tree. To get things started, we assume that your initial tree already has the keys 1 and -1, as illustrated in figure 1.



(a)



(b)

Figure 1: (a) Initial $(2, 3)$-tree. (b) Graph for Prim's Algorithm

Our leaves store only one key (viewed as an "item"), but the keys in internal nodes can duplicate the keys at the leaves. For this part, we only want to see 8 trees, one for each subsequent insertion.

NOTE: Your algorithm for this part must be the same as the one in (iii) next. Be careful! Our insertion algorithm does now always behave the same way as the algorithm in CLRS, for instance.
(iii) For homework, we developed three subroutines which we said can be put together into an insertion algorithm for (a, b)-trees. Let us give names to these routines:

- *SplitRoot*(*oldRoot*) $\rightarrow$ *newRoot*: Assuming *oldRoot* has $b + 1$ children, this subroutine splits *oldRoot* into two halves, and make the two halves children of a new node, which is returned as *newRoot*.
- $MoveKey(u, v) \rightarrow void$: Assuming node u has $b + 1$ children but v is a sibling with $< b$ children, this subroutine moves a key from u into v, fixes up their common parent, and returns a void.
- *Split*$(u, v) \rightarrow$ *parent*: Assuming u has $b + 1$ children and v has b children, we first form the union of u and v and then split the result into 3 nodes which are inserted into their parent node. After fixing up the parent, we return the parent node.

DO NOT RE-DERIVE THESE ROUTINES. Notice that we assumed that an internal node can *temporarily* have $b + 1$ children. State any reasonable assumptions you need for these subroutines. Based on these subroutines, give the pseudo-code for $Insert(k, d, u)$ which inserts an item (k, d) into a (a, b)-tree rooted at u. HINT: it is useful to detect and do something different at nodes whose children are all leaves; call these "leaf-parent" nodes.

SOLUTION:

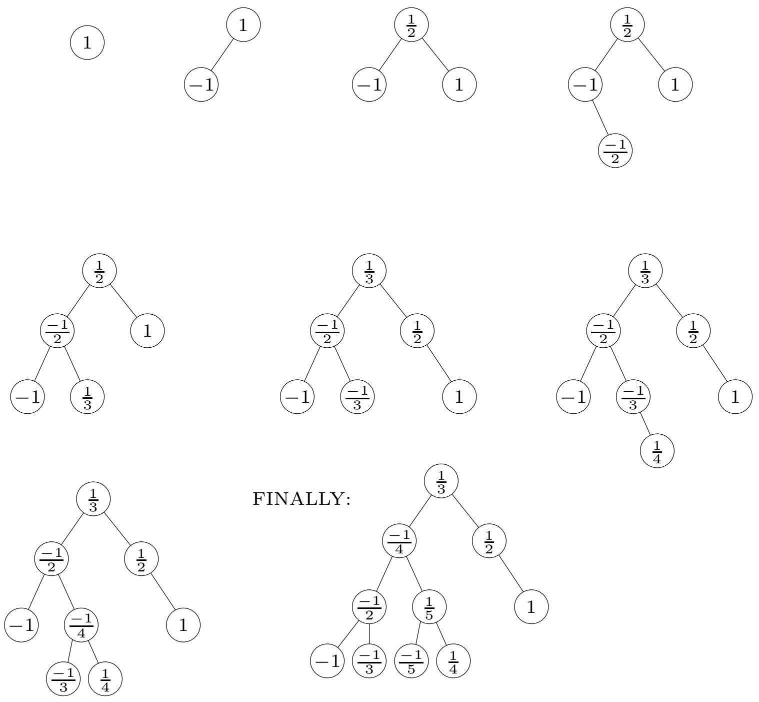


Figure 2: Inserting $1, -1, 1/2, -1/2, , \ldots, 1/5, -1/5$ into an AVL tree.

(i) See Figure 2 for a partial solution.

(ii) See Figure 3 for a partial solution.

PITFALLS: You must be consistent in how keys are duplicated in the internal nodes: if you duplicate keys so that the keys in a left subtree are all $\leq$ the comparison key, then you must make sure that all the right subtree are STRICTLY greater than the comparison key.

(iii) Following the HINT, we assume that we can detect if a node is a leaf-parent.

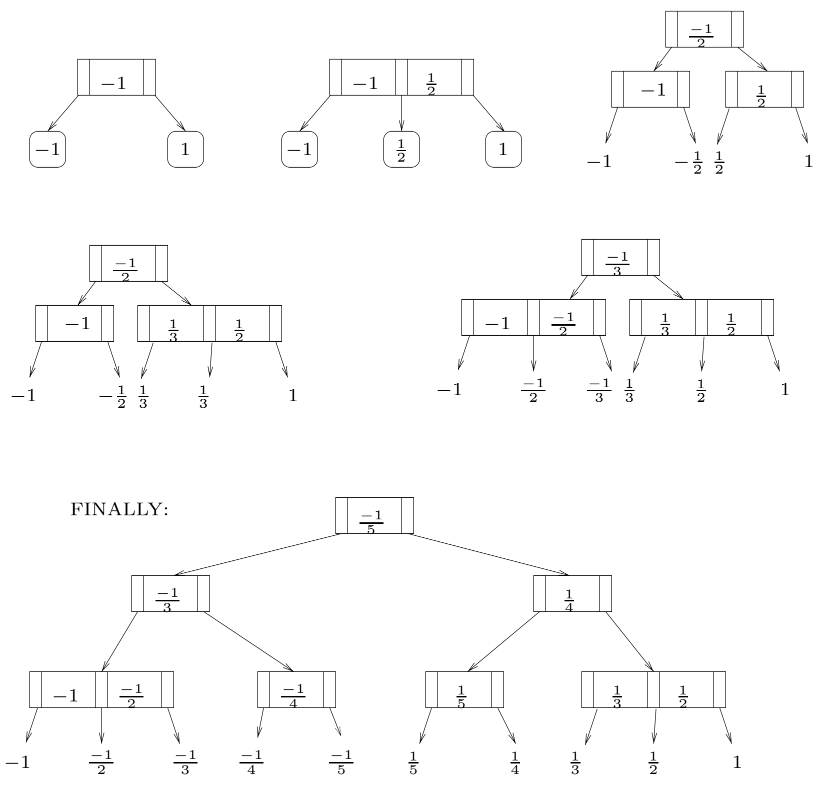


Figure 3: Inserting $1, -1, 1/2, -1/2, , \ldots, 1/5, -1/5$ into an 2-3 tree.

```
INSERT$(k, d, u)$
1      $CURR = u$.
1.1        Assume $CURR$ has this sequence of references and keys:
1.2            $(r_1, k_1, r_2, k_2, \ldots, k_{m-1}, r_m)$.
2      While $CURR$ is not a leaf-parent
2.1        Find the $i$ such that $k_i < k \le k_{i+1}$. (assume $k_0 = -\infty, k_m = \infty$).
2.2        Set $CURR = r_{i+1}$.
3      NOTE: Now $CURR$ is a leaf-parent.
3.1        If $k = k_{i+1}$, return "Insert failed".
3.2        Else, form a new leaf $v$ storing $(k, d)$,
3.3        If $i < m - 1$, insert $(v, k)$ as a child of $CURR$.
3.4        If $i = m - 1$, append $(k', v)$ as the last pair in
               $CURR$, where $k'$ is the key in the leaf $r_m$.
       NOTE: the number $m$ of children of $CURR$ in (1.2) has now been incremented.
4      While $CURR$ has $b + 1$ children:
4.1        If $CURR$ is the root, call $SplitRoot(CURR)$ and break.
4.2        If $CURR$ has a sibling $v$ with $< b$ children, call $MoveKey(CURR, v)$. Break.
4.3        Now, $CURR$ must have a sibling $v$ with $b$ children. Let $CURR = Split(CURR, v)$.
5      Return "Insert successful".
```

3. **(20 points) MST**
Do a hand-simulation of Prim's algorithm on the graph in Figure 1(b), starting your search from vertex 1. We assign numerical costs to vertices in this Figure. The **cost** of edge $(i, j) \in E$ is just $C(i) + C(j)$ where $C(i)$ is the cost of vertex i. Recall from class that this means setting up an array $LC[1..12]$ where the vertex set is $V = \{1, \ldots, 12\}$.

Initially, the array is $LC[1] = 0$ and $LC[i]$ is the weight of the edge $(1, i)$ (weight is infinite if $(1, i)$ is not in the graph). We just display the updated array at each stage.

What is the weight of your MST? What is the MST (you can show this by a figure of the graph)?

Stage	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
1	$\underline{0}$	10	1	8	∞	∞	∞	∞	∞	∞	∞	∞
2			$\underline{1}$	7				3				
3								$\underline{3}$	8			
4				$\underline{7}$	11	11						
5									$\underline{8}$	6		
6						5	2			$\underline{6}$	10	
7					5		$\underline{2}$					
8*						$\underline{5}$						
9					$\underline{5}$							6
10												$\underline{6}$
11*											$\underline{10}$	
12		$\underline{10}$										

The asterisks at steps 8 and 11 indicate that we have a choice about which vertex to pick next.

The weight of the MST is 63 (sum up all the underlined numbers).

The set of MST edges corresponding to the above table is shown in thick lines in Figure 4. By examining this table, you could determine these edges (try it).

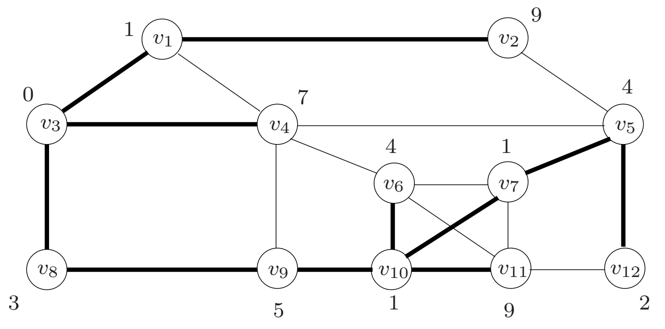


Figure 4: An MST.

4. **(20 points) Recurrences**
Consider the recurrence $T(n) = 10T(n/3) + n$. In homework, we showed by real induction that $T(n) = Cn^\alpha$ eventually, where $\alpha = 3$. That proof will work for any α that is greater than $\log_3(10)$. Now we let $\alpha = \log_3(10)$. Prove by real induction that there exists C and D such that

$$T(n) \le Cn^\alpha - Dn \tag{1}$$

for n large enough. HINT: for the basis, find an interval $[n_0, n_1]$ such that has (at least) these three properties: (i) $1 < n_0 < n_1$ (ii) for any $n \ge n_1$, $n/3$ will fall inside this interval, and (iii) the bound (1) is true for all n inside this interval.

SOLUTION:

First, let us prove the Real Induction Step: we have

$$\begin{array}{rcll} T(n) & = & 10T(n/3) + n & \text{(assumption)} \\ & \le & 10\left[C(n/3)^\alpha - D(n/3)\right] + n & \text{(induction hypothesis)} \\ & = & Cn^\alpha - Dn\left[(10/3) - (1/D)\right] & \\ & \le & Cn^\alpha - Dn & \text{provided } (10/3) - (1/D) \ge 1. \end{array}$$

The last proviso is equivalent to $D \geq 3/7$.

HERE is a better basis: let $T(n) = 0$ for $n < n_0 = 1$. Choose $n_1 = 3$. Then for all $n \in [n_0, n_1)$, we have $T(n) = 10T(n/3) + n = n$. If we choose $D = 3/7$ and $C = 10/7$ then, we have $T(n) = n = n(C - D) < n(Cn^{\alpha-1} - D) \leq Cn^{\alpha} - Dn$. That is, the induction hypothesis is true for $n \in [n_0, n_1)$. This satisfies all the conditions in our HINT. The rest follows by Real Induction.