CIS507: Design & Analysis of Algorithms
Mid-Term, Spring 2012
(VERSION WITH ANSWERS)

Duration: 90 minutes
Total weight: 20%

Student Name: –

Student ID: –

Question	Points Obtained	Points Possible
1		6
2		8
3		4
4		2
Total		20
7 (bonus)		3
Grand Total		23

Cheat Sheet: Master Method

For $T(n) = aT(n/b) + f(n)$, with $a \geq 1$, $b \geq 1$, compare $f(n)$ with $n^{\log_b a}$.

Case	Condition, for $\epsilon > 0$	Solution
1	$f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$	$T(n) = \Theta(n^{\log_b a})$ Number of leafs dominates
2	$f(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n^{\log_b a} \lg n)$ All rows have same asymptotic sum
3	$f(n) = \Omega(n^{\log_b a + \epsilon})$	$T(n) = \Theta(f(n))$ provided that $af(n/b) \leq cf(n)$ for some $c < 1$
2 (general)	$f(n) = \Theta(n^{\log_b a} \lg^k n)$ or some constant $k \geq 0$	$T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ They grow at 'similar' rate

1 Orders of Growth (6 points)

For each pair of functions below, indicate whether $f(n) = \mathcal{O}(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. Assume $n \geq 1$, $f(n) \geq 1$ and $g(n) \geq 1$.

1. **(1 point)** $f(n) = (n+5)^2$ versus $g(n) = n^2$
2. **(1 point)** $f(n) = n!$ versus $g(n) = n^n$
3. **(1 point)** $f(n) = \log(n!)$ versus $g(n) = n \log n$
4. **(1 point)** $f(n) = 2^n$ versus $g(n) = 2^{n^2}$
5. **(1 point)** $f(n) = 2^{2^n}$ versus $g(n) = 2^{n^2}$
6. **(1 point)** $f(n) = (\sqrt{2})^{\lg n}$ versus $g(n) = \sqrt{n}$. (lg is base 2 logarithm)

Grading notes:

- If $f(n) = \Theta(g(n))$, writing $f(n) = \mathcal{O}(g(n))$ or $f(n) = \Omega(g(n))$ gets 0.25.
- If $f(n) = \mathcal{O}(g(n))$ or $f(n) = \Omega(g(n))$, writing $f(n) = \Theta(g(n))$ gets 0.

ANSWER:

1. $f(n) = \Theta(g(n))$. Observe that $f(n) = (n+5)^2 = n^2 + 10n + 25$. Now:
   - Choose $c_1 = 1$, then $n^2 + 10n + 25 \geq n^2$ for all $n \geq 1$. Therefore $f(n) = \Omega(g(n))$.
   - Choose $c_2 = 10$, then $n^2 + 10n + 25 \leq 10n^2$ for all $n \geq 10$. Therefore $f(n) = \mathcal{O}(g(n))$.
2. $f(n) = \mathcal{O}(g(n))$. Because $n! = n(n-1)(n-2)\ldots 1 \leq n^n$.
3. $f(n) = \Theta(g(n))$. As found above, $n! \leq n^n$. Now, take logs of both sides: $\log(n!) \leq \log(n^n) = n \log n$. Therefore, with $n_0 = c = 1$,we have $\log(n!) = \mathcal{O}(n \log n)$. Now, to see that $n! = \Omega(n \log n)$, observe that $\log(n!) =$

$$\begin{aligned}
\log(1.2\ldots n) &= \log(1) + \log(2) + \cdots + \log(\frac{n}{2}) + \cdots + \log(n) \\
&\geq \log(\frac{n}{2}) + \cdots + \log(n) \\
&\geq \log(\frac{n}{2}) + \cdots + \log(\frac{n}{2}) \\
&= \frac{n}{2}\log(\frac{n}{2}) = \frac{n}{2}\log(n-1) = \frac{n}{2}\log(n) - \frac{n}{2}
\end{aligned}$$

Now we show that $\frac{n}{2}\log(n) - \frac{n}{2} \geq cn\log n$ for $c = \frac{1}{4}$ for all $n \geq 4$. If $n \geq 4$ then:

$$\begin{aligned} \log n \geq 2 \quad &\Rightarrow \quad \frac{1}{4}\log n \geq \frac{1}{2} \\ &\Rightarrow \quad \frac{n}{4}\log n \geq \frac{n}{2} \\ &\Rightarrow \quad \frac{n}{4}\log n - \frac{n}{2} \geq 0 \\ &\Rightarrow \quad \frac{n}{2}\log n - \frac{n}{2} \geq \frac{n}{2}\log n \end{aligned}$$

Therefore, $\log(n!) \geq \frac{n}{2}\log(n) - \frac{n}{2} \geq \frac{1}{4}n\log n$ for all $n \geq 4$. Hence $n! = \Omega(n\log n)$.

4. $f(n) = \mathcal{O}(g(n))$. Taking logarithm of both, we compare n with n^2.
5. $f(n) = \Omega(g(n))$. Taking logarithm of both, we compare 2^n versus n^2.
6. $f(n) = \Theta(g(n))$. Since: $(\sqrt{2})^{\lg n} = n^{\lg\sqrt{2}} = n^{\lg 2^{1/2}} = n^{\frac{1}{2}\lg 2} = n^{\frac{1}{2}} = \sqrt{n}$

2 True or False (8 points)

1. **(1 point)** Dijkstra's algorithm works on any graph without negative weight cycles.
2. **(1 point)** If an in-place sorting algorithm is given a sorted array, it will always output an unchanged array.
3. **(1 point)** There exists a comparison sort of 5 numbers that uses at most 6 comparisons in the worst case.
4. **(1 point)** Given a hash table with more slots than keys, and collision resolution by chaining, the worst case running time of a lookup is constant time.
5. **(1 point)** Linear probing satisfies the assumption of uniform hashing.
6. **(1 point)** Let P be a shortest path from some vertex s to some other vertex t in a graph. If the weight of each edge in the graph is increased by one, P will still be a shortest path from s to t.
7. **(1 point)** A graph can have more than one minimum spanning tree.
8. **(1 point)** Multiplying all edge weights by a positive number might change the shortest path between two vertices u and v.

ANSWER:

1. False. A single negative edge makes false the assumption that when you expand a node, you have already found the shortest path to that node.
2. False. In-place just means that it only uses a constant amount of extra memory. Stable would mean it would leave the array unchanged.
3. False. The number of leaves of a decision tree which sorts 5 numbers is 5! and the height of the tree is at least lg(5!). Since $5! = 120$, $2^6 = 64$, and $2^7 = 128$, we have $6 < \lg(5!) < 7$. Thus at least 7 comparisons are required.
4. False. In the worst case we get unlucky and all the keys hash to the same slot for $\Theta(n)$.
5. False. The second probe can be determined exactly from the first probe, wihle uniform hashing says that the entire permutation of probes is random.
6. False. Suppose $w(s, v_1) = 1$, $w(v_1, v_2) = 1$, $w(v_2, t) = 1$, and $w(s, t) = 4$. The shortest path would change if we add 1 to each edge weight.
7. True. As an example, consider a graph with all edge weights of 1, and one cycle. Deleting any edge on that cycle yields a MST.
8. False. Simply, we get the same MST, but with weight $c.w(T)$, where $w(T)$ is the weight of the original MST.

3 Multiple Choice (4 points)

For each of the following, circle the correct answer.

1. **(1 point)** Suppose instead of dividing in half at each step of the mergesort, you divide into thirds, sort each third, and finally combine all of them using a three way merge. What is the overall running time of this algorithm? (*Hint:* Note that the merge step can still be implemented in $\mathcal{O}(n)$ time.)
   (i) $\mathcal{O}(n^3)$ (ii) $\mathcal{O}(n^2 \lg n)$ (iii) $\mathcal{O}(n \lg n)$ (iv) $\mathcal{O}(n^3 \lg n)$

2. **(1 point)** Suppose you are given k sorted arrays, each with n elements, and you want to combine them into a single array of kn elements. Consider the following approach. Using the merge subroutine, you merge the first 2 arrays, then merge the 3rd given array with this merged version of the first two arrays, and so on until you merge in the final (kth) input array. What is the time taken for this strategy, as a function of k and n?
   (i) $\Theta(n^2)$ (ii) $\Theta(nk^2)$ (iii) $\Theta(kn^2)$ (iv) $\Theta(kn)$

3. **(1 point)** Suppose the running time of an algorithm follows the recurrence $T(n) = 9T(\frac{n}{3}) + n^2$. What is the asymptotic running time?
   (i) $\Theta(n \log n)$ (ii) $\Theta(n^2 \log^2 n)$ (iii) $\Theta(n^2 \log n)$ (iv) $\Theta(n^2)$

4. **(1 point)** Let h_1, h_2, h_3, h_4 be hash functions from the set of keys $U = \{0, 1, 2, 3, 4\}$ to table $T = \{0, 1\}$ defined as follows:

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	1	0	1	0
1	0	1	0	1
2	1	0	0	1
3	0	1	1	0
4	1	1	0	0

   Which of the following sets H is universal? (choose one only)
   (i) $\{h_1, h_2, h_3, h_4\}$ (ii) $\{h_1, h_3\}$ (iii) $\{h_2\}$ (iv) $\{h_4\}$

ANSWER:

1. (iii) $\mathcal{O}(n \lg n)$. There is still a logarithmic number of levels, and the overall amount of work at each level is still linear.

2. (ii) $\Theta(nk^2)$. First merge costs $2n$, second costs $3n$, etc. Total is $2n + 3n + \cdots + kn = n(2 + 3 + \cdots + k) = n(\frac{k(k-1)}{2} - 1) = \Theta(nk^2)$

3. (iii) $\Theta(n^2 \log n)$. This is case 2 of the master theorem.

4. (i) $\{h_1, h_2, h_3, h_4\}$. For any pair of keys x, y, the number of functions that collide must be $|\{h \in H : h(x) = h(y)\}| \leq \frac{|H|}{|T|}$. For $H = \{h_1, h_2, h_3, h_4\}$, $\frac{|H|}{|T|} = \frac{4}{2} = 2$, and for any pair of keys, at most 2 functions collide (e.g. for $x = 0$ and $y = 2$, only h_1 and h_2 collide). This is not true for $H = \{h_1, h3\}$ for example. $\frac{|H|}{|T|} = \frac{2}{2} = 1$, but for $x = 2$, $y = 4$, two functions collide.

4 Trace Radix Sort (2 points)

Radix sort the following list of integers in base 10 (smallest at top, largest at bottom). Show the resulting order after each run of the stable sorting subroutine.

Original List	First sort	Second sort	Third sort
583			
625			
682			
243			
745			
522			

Grading: -1 for minor errors. -2 for major error.

ANSWER:

Original List	First sort	Second sort	Third sort
583	682	522	243
625	522	625	522
682	583	243	583
243	243	745	625
745	625	682	682
522	745	583	745

5 BONUS: Random Coloring (3 points)

You are given a graph $G = (V, E)$ with nodes (vertices) V and edges E between them. You have 3 colored pens, and you are asked to color all nodes.

For every pair of nodes a and b such that $(a, b) \in E$, the edge $e = (a, b)$ is *satisfied* if a and b have different colors. For each edge that is satisfied, you will be given \$1. Theoretically, your maximum reward is at most $\$|E|$.

You like money, but you are also lazy! So you decide to color each node randomly (i.e. for each node, choose a color uniformly at random, independent of the colors of other nodes). Compute how much you expect to make as a function of $|E|$. Show your full derivation.

Hint: Use indicator random variables over edges.

ANSWER:

Define indicator random variable X_e for each edge $e = (a, b)$, which is 1 if a and b have different colors.

$$X_e = \begin{cases} 1 & \text{if nodes } a \text{ and } b \text{ have different colors} \\ 0 & \text{otherwise} \end{cases}$$

Then, for any edge, there are 9 ways to color its two ends, each of which appears with the same probability, and 3 of them will deduct your reward. So we expect $\frac{6}{9} = \frac{2}{3}$ to be satisfied on average:

$$E[X_e] = Pr\{e \text{ is satisfied}\} = \frac{6}{9} = \frac{2}{3}$$

Let Y be a random variable denoting the number of satisfied edges. We have:

$$Y = \sum_{e \in E} X_e$$

Take expectation of both sides:

$$E[Y] = E\left[\sum_{e \in E} X_e\right]$$

By linearity of expectation:

$$E[Y] = \sum_{e \in E} E\big[X_e\big] = \sum_{e \in E} \frac{2}{3} = \frac{2}{3}|E|$$