

## CS 146 Final Exam 2 Solutions

### 1. Analysis of non-recursive algorithms

Given an array of integers  $x$  containing  $N$  entries, we wish to find the two entries  $a = x[i]$  and  $b = x[j]$  such that  $|a-b|$  is as large as possible. Assume  $N > 1$ .

(a) write code that correctly performs this task and returns an `int[]` containing  $i$  and  $j$ . (Not  $a$  and  $b$ .) Solve this problem as efficiently as you can.

*Answer. There are two solutions. One is a double for-loop searching directly for the desired numbers. The other finds the minimum and then the maximum and subtracts them. Here are both solutions.*

```
int[] maxDiff(int [] x)
{ int[] ans = new int[2];
  int i,j,n=ans.length;
  int best = 0;
  ans[0] = 0; ans[1] = 1;
  for(i=0;i<n;i++) for(j=i+1;j<n;j++)
  { if(Math.abs(x[i]-x[j]) > best)
    { ans[0] = i; ans[1] = j;
    }
  }
  return ans;
}
```

```
int[] maxDiff(int[] x)
{ int[] ans = new int[2];
  int i;
  int big = x[0];
  int small = x[0];
  int n = x.length;
  for(i=0;i<n;i++)
  { if (x[i] > big)
    {big = x[i]; ans[1] = i;}
    if (x[i] < small)
    {small = x[i]; ans[0] = i;}
  }
  return ans;
}
```

(b) Give the running time of this code in terms of  $N$  using  $\Theta$  notation.

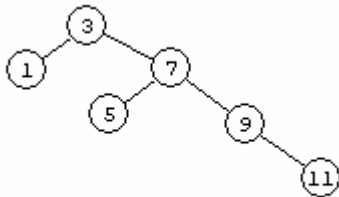
*Answer: The first one is  $O(N^2)$  and the second one is  $O(N)$ .*

2. Indicate, for each pair of expressions (A,B) in the table below, whether A is  $O$ , or  $\Theta$  of B. Write “Yes” or “No” in each blank box of the table. [You will only receive points if you answer more than 50% correct, since you can get 50% correct by guessing.]

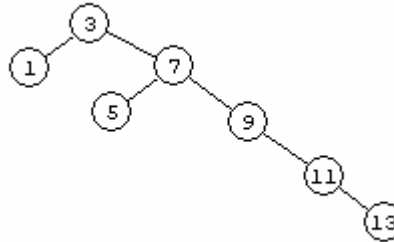
A	B	$O$	$\Theta$
$2^n$	$n^3$	No	No
$\sqrt{n}$	$n^3 + \sqrt{n}$	Yes	No
$\sqrt{n}$	$\lg n$	No	No
$\sqrt{n}$	$\lg n + \sqrt{n}$	Yes	Yes

3. *Red-black trees.* For each of the following two trees, either indicate how to color the nodes red and black (use R and B to label the nodes) to make the tree a red-black tree, or explain why that is not possible.

(a)



(b)



For (a), color nodes 7 and 11 red. Then the black height of the root will be 2 on all branches, e.g. 1-nil, and 9-nil on the path 3-7-9-nil, and 9-nil on path 3-7-9-11-nil.

For (b) it's not possible, as the black-height of 3 has to be at most 2 on the left branch, but must be at least 3 on the right branch.

4. *Linked lists.* In mathematics, a *polynomial* is a sum of *monomials*, where a *monomial* is a term like  $5x^3$ , with a coefficient (5 in the example) and an exponent (3 in the example). The  $x$  is just a place-holder (if we work with polynomials of one variable) and the  $+$  can just as well be a linked-list arrow, so we can use

```
class Polynomial {
    int coef;
    int exp;
    Polynomial next;
}
```

to represent a polynomial. Thus  $x^{119} - 1$  needs only a few bytes (two nodes) instead of the hundreds it would need in an array representation, where a polynomial is represented as an array of coefficients, with the  $k$ -th entry representing the coefficient of  $x^k$ . We also require that the terms be in ascending order of exponent, so the lowest exponent is in the first node of the list.

Write the code that adds two polynomials and returns the sum in a new list, without disturbing either of the two inputs.

```
static Polynomial add(Polynomial x, Polynomial y)
{ Polynomial marker1=x,marker2=y;
  Polynomial ans, p;
  if(x==null && y==null) return null;
  ans = new Polynomial();
  for(p=ans; marker1!=null || marker2 != null;
    p.next = new Polynomial(),p = p.next)
  { if(marker1==null || (marker2 != null && marker1.exp < marker2.exp))
    { p.coef = marker2.coef;
      p.exp = marker2.exp;
      marker2=marker2.next;
    }
    else if (marker2==null || (marker1 != null && marker2.exp < marker1.exp))
    { p.coef = marker1.coef;
      p.exp = marker2.exp;
      marker1=marker1.next;
    }
    else if(marker1.exp == marker2.exp)
    { p.exp = marker1.exp;
      p.coef = marker1.coef + marker2.coef;
      marker1=marker1.next;
      marker2=marker2.next;
    }
  }
  return ans; } // last brace fits on the page if I put it on this line
```

For completeness I also present a recursive solution of problem 4:

```
Polynomial copy()
// return a copy of this
{ Polynomial ans = new Polynomial();
  ans.exp = exp; ans.coef = coef;
  ans.next = next.copy();
  return ans;
}
static Polynomial addRec(Polynomial x, Polynomial y)
{ if(x==null && y==null) return null;
  if(x==null) return addRec(y,x);
  if(y==null) return x.copy();
  Polynomial ans = new Polynomial();
  if(x.exp == y.exp)
    { ans.exp = x.exp;
      ans.coef = x.coef + y.coef;
      ans.next = addRec(x.next,y.next);
      return ans;
    }
  if(x.exp < y.exp)
    { ans.exp = x.exp;
      ans.coef = x.coef;
      ans.next = addRec(x.next,y);
      return ans;
    }
  return addRec(y,x);
}
```

5. Analyzing recursive code using recurrence relations.

Here is an algorithm to find all the factors of a positive integer  $N$ , and return a string listing the factors.

```
String factor(int N)
{
    int k;
    if(N==1) return ""; // 1 has no factors
    int bound = (int) Math.sqrt(N+0.0001);
    for(k=2;k<=bound;k++)
        if(N % k == 0) // found one factor, namely k
            return k + " " + factor(N/k);
    return "" + N; // N was prime
}
```

(a) Write a recursion relation to describe the worst-case running time . You might find this easier to do in terms of  $m$ , where  $N$  has  $m$  binary digits. But I will accept a correct answer either in terms of  $N$  or  $m$ .

$$T(N) \leq O(\sqrt{N}) + T(N/2) \quad ; \quad \text{or} \quad T(m) = O(2^{m/2}) + T(m-1)$$

So to get the bound we solve the relation with = instead of  $\leq$

(b) Solve that recursion relation .  $T(m) = O(2^{m/2})$

which we can see this way:  $T(m) = c(2^{m/2} + 2^{m/4} + 2^{m/8} \dots) < c 2^{m/2+1} = O(2^{m/2})$   
since the sum on the left is less than the sum of all the powers of 2 up to the one indicated, and the sum of all powers of 2 up to a given one is one less than the next power of 2.

Or,  $T(N) = O(\sqrt{N})$

which we see as  $T(N) = c(N^{1/2} + N^{1/4} + N^{1/8} + \dots + 1) \leq 2cN^{1/2}$

6. *Binary Search Trees.*

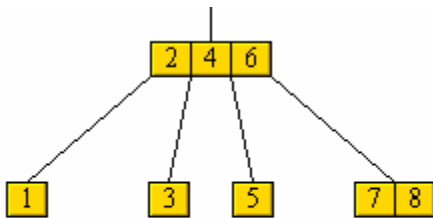
We can sort a given set of  $n$  numbers by first building a binary search tree containing these numbers (using TREE-INSERT repeatedly to insert the numbers one by one and then printing the numbers by an inorder tree walk. What are the worst-case and best-case running times for this sorting algorithm?

Worst-case refers to the situation where the inserts repeatedly occur down the same branch. Best-case refers, let us say, to the situation where the final tree's longest branch is no more than 10 times its shortest branch. (Any constant would do in place of 10.) Give your answer using O-notation.

(a) *Best case answer*  $T(n) = O(n \lg n)$  since each insertion takes at most  $\lg n$  and the final tree traversal takes  $O(n)$ .

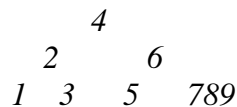
(b) *Worst case answer*  $O(n^2)$  since the worst case is, for example, when the input is  $1, 2, \dots, n$ . Then the insertion takes  $1 + 2 + 3 + \dots + n = n(n+1)/2 = O(n^2)$  steps.

7. *B-tree insertion and deletion.* A new key 9 is to be inserted into the following B-tree. This B-tree has 2,3, or 4 children per internal node.



(a) Draw a similar diagram showing the B-tree after the insertion of 9. Use the one-pass insertion algorithm, i.e. the one given in your textbook

*Answer: The root has to split, so the new root is 4. The new tree looks like this, sorry I can't make it look as nice as the problem:*



8. *Merge sort*. The following array is to be sorted:

10	2	9	15	8	17	14	5	3	19	13	22	7	11	6	12
----	---	---	----	---	----	----	---	---	----	----	----	---	----	---	----

What will the array look like immediately after the return from the first recursive call?  
*The first half is sorted:*

2	5	8	9	10	14	15	17	3	19	13	22	7	11	6	12
---	---	---	---	----	----	----	----	---	----	----	----	---	----	---	----

9. *Quick sort*. The following array is to be sorted (it's the same as the one above):

10	2	9	15	8	17	14	5	3	19	13	22	7	11	6	12
----	---	---	----	---	----	----	---	---	----	----	----	---	----	---	----

(a) if *quickSort* is used to sort this array, what will the array look like just before the first recursive call?

10	2	9	8	5	3	7	11	6	12	13	22	14	15	17	19
----	---	---	---	---	---	---	----	---	----	----	----	----	----	----	----

(b) What will the array look like just after the first recursive call?

2	2	5	6	7	8	9	10	11	12	13	22	14	15	7	19
---	---	---	---	---	---	---	----	----	----	----	----	----	----	---	----

10. *Heaps*

(a) A certain heap contains 1200 nodes. What are the lengths of the longest and shortest paths in that heap? (For example, if a heap has 3 nodes, arranged with one root and two children, the longest and shortest paths in that heap both have length 1. The example is meant to illustrate that we count parent-child pairs, not nodes, in counting the length of a path.)

*Shortest 9          Longest 10*

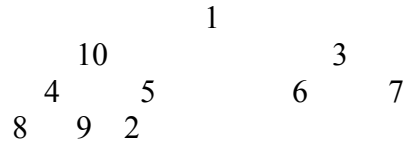
*When all the paths have length 9 then the heap contains 1023 nodes just as when all the paths are length 2 it contains 7 nodes. So then nodes 1024 to 1200 have paths of length 10 but then there are still a lot of length 9 paths left.*

(b) In a certain heap, there are three nodes with data 12, 15, and 18. One of them is the parent of the other two. Which one is definitely *not* the parent? *15. In a min-heap, 12 will be the parent, and in a max-heap, 18 will be the parent.*

11. *Heap algorithms.* The following array represents a binary tree.

1	10	3	4	5	6	7	8	9	2						
---	----	---	---	---	---	---	---	---	---	--	--	--	--	--	--

(a) Using the convention we used for heaps, draw a picture of this tree.



(b) After calling MIN-HEAPIFY, the resulting array will be a min-heap. Show it in array form:

1	2	3	4	5	6	7	8	9	10						
---	---	---	---	---	---	---	---	---	----	--	--	--	--	--	--

12. *Recurrence relations.* If we multiply two matrices by dividing each matrix into four equal-sized submatrices and multiplying the submatrices recursively, we get the recurrence relation  $T(n) = 8 T(n/2) + \Theta(n^2)$ .

(a) Solve this recurrence relation. Is it slower, or faster, or the same speed (up to a constant factor) as the usual way of multiplying matrices (which involves three nested for-loops)?

*Answer:*  $T(n) = \Theta(n^3)$ .

*and it is the same speed. See part b for the explanation.*

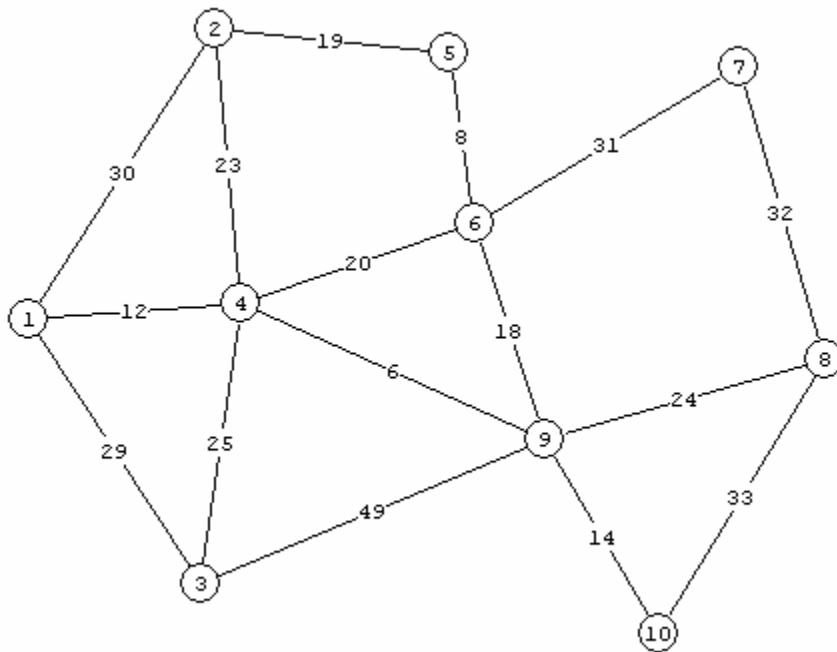
(b) There is a clever way to multiply matrices using only 7 recursive calls, so we get the recurrence relation  $T(n) = 7 T(n/2) + \Theta(n^2)$ . Solve this recurrence relation. Is it faster or slower (for large enough  $n$ ) than the usual way of multiplying matrices?

*Answer:*  $T(n) = \Theta(n^{\lg 7})$ .

*and it is faster, since  $\lg 8$  is 3, so  $\lg 7$  is less than 3 (actually it's 2.81).*

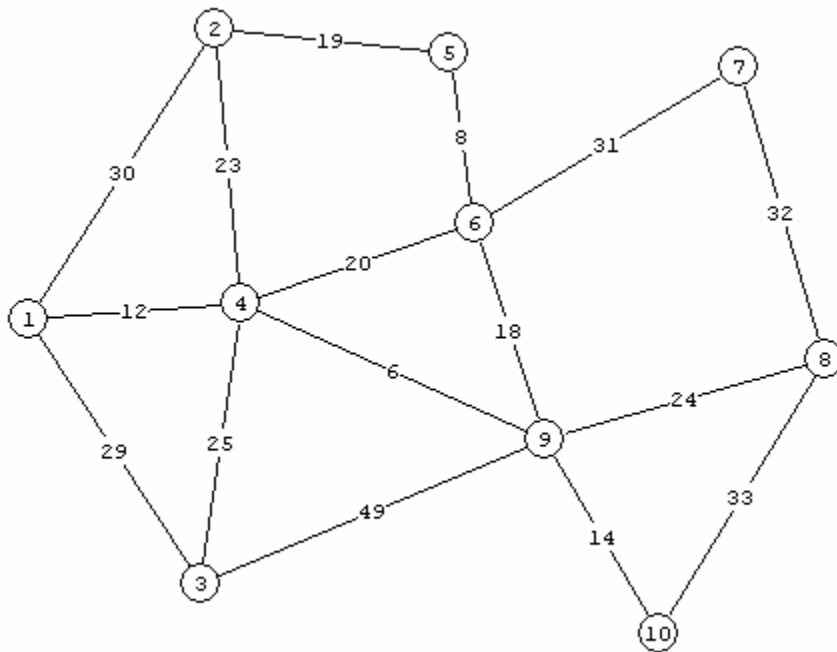
*At the  $k$ -th level of the recursion tree, we get  $7^k$  nodes each contributing  $(n/2^k)^2$ , so the total of the  $k$ -th level is  $n^2 (7/4)^k$ . There are  $\lg n$  levels so we have  $n^2$  times the sum of  $(7/4)^k$  from  $k=1$  to  $\lg n$ . That sum is  $O((7/4)^{\lg n}) = O(n^{\lg(7/4)}) = O(\Theta(n^{\lg 7 - \lg 4})) = \Theta(n^{\lg 7 - 2})$ . So multiplying by  $n^2$  we get the stated answer. For part (a) we have 8 instead of 7, and  $\lg 8$  is 3.*

13. Execute Prim's algorithm to find a minimal spanning tree in the following graph, starting from vertex 6. List the edges by weight in the order they are added to the minimal spanning tree. The answer should be a list of numbers.



*Answer : 8, 18, 6, 12, 14, 19, 24, 25, 31*

14. Execute Kruskal's algorithm to find a minimal spanning tree in the following graph (which is, incidentally, the same graph as in the previous problem). List the edges by weight in the order they are added. The answer should be a list of numbers.



*Answer: 6, 8, 12, 14, 18, 19, 24, 25, 31. Correctness check: this should be (and is) the same edges as when we used Prim's algorithm, but in sorted order.*

15. If we run Dijkstra's algorithm on the graph in the previous problem, starting from node 7,

(a) how many nodes will be removed from the queue before every node gets its correct distance from node 7 calculated?

*Answer [just one number] : 6*

(b) List the nodes in the order of their removal from the queue.

*Answer [a list of numbers]: 7, 6, 5, 9, 4, 2, 1, 10, 3 or with 10 and 1 in the other order.*

16. Needleman-Wunsch.

(a) What are the asymptotic time and space requirements of the Needleman-Wunsch algorithm? Express your answers in terms of the maximum length  $n$  of the two input strings, using big-O notation.

Time:  $O(n^2)$

Space:  $O(n^2)$

(b) Execute the Needleman-Wunsch algorithm to find the best alignment of MARY and SAY, scoring exact matches as 1 and non-matches as 0, with a gap penalty of 1 (i.e. matching a letter with a gap scores -1). Use the following table to execute the algorithm, showing the predecessor arrows as was done in class.

	-	M	A	R	Y
-	0	-1L	-2L	-3L	-4L
S	-1U	0D	-1D	-2D	-3D
A	-2U	-1D	1D	0L	-1L
Y	-3U	-2D	0U	1D	1D

The best alignment is MARY  
SA - Y

and the score of that alignment is 1 (two matches = 2 minus one for a gap )

17. In your programming assignment *Google*,

(a) The pages are given by their names, which are strings. How are they indexed so that one can rapidly find the  $n$  such that a given page is  $pages[n]$  ?

*A hash table is used.*

(b) In the B-matrix, what does an entry in row 5, column 7 represent?

*A link from pages[7] to pages[5]*

(c) The main loop updates the rank of each page. What is the way that the array *ranks* is updated (in each iteration of the main loop)?

*The rank array, considered as a column matrix, is multiplied by the B-matrix, and the result is the updated ranks array.*

18. In your programming assignment *MazePath*,

(a) What algorithm is used? *Dijkstra*

(b) *Multiple choice*: You have to use that algorithm because

-- ~~it is more efficient than the ones we learned earlier, or~~  
-- **the ones we learned earlier will give wrong answers on *MazePath***

19. *Graphs and their representations*

Consider a delivery service such as FedEx. It is concerned with the cost and the time required for verified delivery of a package from A to B. Generally a customer will want to know the options. For example, recently I had to ship a CD to South Africa, and I could choose between \$55 for five-day delivery and \$75 for three-day delivery.

(a) What are the vertices of the relevant graph? *FedEx offices and distribution centers.*

(b) What results along these lines could be obtained by direct application of algorithms we studied in this course?

*You could find the **cheapest way***

*or the **fastest way***

*by the direct application of **Dijkstra's algorithm***

(c) If you were hired by FedEx to write a program that could be used to generate the options for a customer, how would you proceed? (hopefully you would use some things you learned in this course.)

*Given a cost  $M$ , we can modify Dijkstra so that it uses times for the weights, but also keeps track of the cost of getting to a node in the time stored at that node, and considers it impossible if the cost reaches or exceeds  $M$ . Then only paths of cost  $< M$  will be found. Now, first find the fastest way by unmodified Dijkstra, and let its cost be  $M$ . Then run Dijkstra again to find a slower but cheaper way than the first way. Repeat this, generating successively slower and cheaper routes, until you reach the cheapest one (found by using Dijkstra with costs instead of times for weight).*