Algorithms Homework Problems.

Additional problems may be announced in class or posted.

Due Dates are posted on the class web site:

`www.unf.edu/~wkloster/4400/4400.html`

Preliminary: You are expected to read and be familiar with the material in the appendix, part B. In addition, you should be familiar with concepts such as logarithms, floor and ceiling functions, mod, and binary arithmetic.

"Recommended" problems will not be collected by are listed to encourage you to work them (perhaps as a review for the exam).

Exam 1 material:

0. Prove by induction that

$$\sum_{k=1}^{n} 2 * 3^{k-1} = 3^n - 1$$

1. 1.2-2
2. 1.2-3
Recommended: 1-1, 2.3-7, A1-1
3. 3.1-4
4. 3.2-1
5. 3-4a
6. 3-4e
7. 3-4b
8. 3-3 (Consider only those functions on the last row)
9. True or false:
a. $2^n + 2^{n+1} \in O(3^n)$
b. $n! \in O((n+1)!)$
c. $(n+1)! \in O(n!)$
d. $n^2 + n^3 + 5675 \in O(n^3 - n^2)$

10. Determine whether f is $o(g), O(g)$, or $\Omega(g)$
f , g
a. $4n^2 + 8n$, $3n^3$
b. $n^2 + 3n$, $\frac{n^2}{2}$
c. $n \log n$, $n \log^2 n$
d. n, $\log^* n$
e. $\log n$, $n^{\frac{1}{100}}$
f. n^n, $n!$
g. $n^{\frac{1}{2}}$, $\log \log n$

11. A.1-1
12. Prove:

$$\sum_{k=1}^{n} 3 * k^2 = \frac{n(n+1)(2n+1)}{2}$$

13. A.2-1 [this one is hard!)
14. A.2-4

15. 4.1-1 (ignore ceiling)
16. 4.1-2 (hard!) Hint: ignore floor, use induction to show is $O(n \log n)$
17. 4.2-1 (hard!) (you can use Master theorem or if you feel brave, try unrolling and find a summation to describe what happens)
18. 4.3-1
19. 4-1 a, b, c, d
20. 4-4 a, e (for part e, estimate upper bound) (bonus: try g, h, 4-1h)

21. Express the running time of the following:

```
function foo(input: a: array[1..n] of integer; output: b: integer);

i:=1 while i < n/2

  if a[i] < a[2i] then
     i:=ceiling(3i/2)

  i:+i+1;
end while;

return a[1];

end;
```

22. Express the running time of the following:

```
function foo(input: a: array[1..n] of integer; output: b:
array[1..n] of integer);

if n=1 then return a;

swap(a[1], a[n]);
if a[1] < a[2]
   concatenate(foo(a[1..n-1]), foo(a[1]));

return a; end;
```

23. Bonus: Express the running time of the following:

```
function foo(input: a: array[1..n] of integer; output: b: integer);

int x=1;
if n=1 then return a;
i=1;
while i <= n do
  x=foo(a[1..n/2]);
```

```
  i:=i+1;
end while;

return i+x;
end;
```

24. Solve (23) if we replace the "while" loop with "for i=1 to 3"

25. Give a good bound for:

$$\sum_{i=0}^{n} 4^i$$

Exam 2 material:
1. 6.1-3, 6.1-4, 6.1-5, 6.1-6
2. 6.2-1
3. 6.3-1
4a. 6.4-1, 4b. 6.4-2 (bonus)
5. 6-1b, 6-2 a
Recommended: 7.2-1, 7.2-2, 7.2-3
6. 8.2-4
7. 9-1
8. 9.3-1 (Write the recurrence relations for each, for bonus, solve the resulting recurrence relations)
9. 9.3-6 (Hint: Assume k is a power of 2 and find the appropriate order statistics). This is hard!
10. 16.2-3
11. 16.2-4
12. We have n people and n pairs of skis. Ideally, we would like each person's skis to be the same length as the person's height. Design an algorithm to minimize the sum of differences between the people's heights and the length of the pair of skis assigned to them. Hint: First show that the following greedy strategy does not give the optimal solution:

```
repeat
   assign the current person the pair of skis closest to his/her height
until no more people
```

13. 22.1-6
14. 22.2-1
15. 22.3-4
16.a 23.1-1

b. Run Prim's algorithm starting from vertex c) using the graph on page 571
17. (bonus) 23-1.c [Hint: read the (a) part of the question], 23-1.d
18. (bonus) 16.1-2
19. 24.3-1
20. 26.2-1, 26.2-2, 26.3-1
21. Given an array of n integers and an integer x, give an $O(n \log n)$ algorithm to determine if there are two elements from the array whose sum is x.

22. Modify Dijkstra's algorithm to find a path between u and v that minimizes the maximum edges weight on the path (this type of problem is known as a "bottleneck" problem).
23. Input two separate lists of pairs (a, b) where each a, b is an integer between 1 and n. No pair appears more than once in a list. In linear time, output the pairs that occur in both lists.
24. Write an algorithm that takes a queue of n integers and using only one stack (and possibly a few other integer variables, but no arrays or other data structures) places the elements in the queue in ascending order. Analyze the running time of your algorithm.
25. (Bonus) 24.2-1, 24.3-4, 24.3-5
26. (Bonus): Let A, B be two arrays of n integers. Let C be the array $\{a + b | a \in A, b \in B\}$. Note that C has $O(n^2)$ elements. Write an $O(n \log n)$ algorithm to print the n largest elements in C. [Interesting Fact: the problem can actually be solved in $O(n)$ time Frederickson and Johnson, SIAM Journal on Computing, 1984, assuming the arrays are already sorted.]

Final exam material:

0. What is the running time of the nearest-neighbor algorithm for TSP? What is the running time of the greedy graph coloring algorithm (do for adjacency list and for adjacency matrix)?

0.1 Determine if graph G contains a uv path, all of whose edge weights are equal.

0.2 (bonus) Suppose we want to find the uv path such that the average edge weight on the path is minimized. Is there a polynomial time algorithm? How long might such a path be?

0.3 Show (by an example) that Dijkstra's algorithm (when you change MIN to MAX and $<$ to $>$) fails to compute longest paths.

1. 15.2-1
2. 15-4
3. Given two text files A and B, write an efficient algorithm to find the longest text sequence common to both A and B. You should be able to find an algorithm running in $O(n^3)$ time, where n is the maximum of the lengths of A and B. $O(n^2)$ is possible!
4. An independent set in a graph is a subset of the vertices, no two of which are adjacent. The maximum independent set is the largest such subset. A Maximal Independent Set of a graph $G = (V, E)$ is an independent set M such that the addition to M of any vertex in $V - M$ creates a non-independent set. Write an efficient algorithm to find a maximal independent set in a graph. Demonstrate a graph on which your algorithm's maximal independent set is not the maximum independent set.
5. Prove that the size of a maximum matching in a bipartite graph is equal to the number of vertices in minimum vertex cover of the same graph.
6. 26.2-9
7. (bonus) Can the following be bounded by a polynomial function?

a) T(n) = T(n-1) + T(n/2) + 1; T(1) = 1.
b) T(n) = T(n-1) + T(n/2) + n; T(1) = 1.

8. 34.2-2
9. 34-3a
10. Give a polynomial time algorithm the following: "Does G have a dominating set of size at most 3?" [A dominating set D is a subset of vertices such that every vertex in G is either in D or has at least one neighbor in D]
11. 35.1-4

12. Recall the all-pairs shortest path problem (see section 25.1). Note that during the computation of matrix L^{n-1}, one can detect if there is a "path" of length $n-1$ between u and v (by seeing if there is a uw path of length $n-1-c$ and a wv path of length c). Why is this not a polynomial time algorithm for Hamiltonian Path? In other words, why can we not easily modify Floyd-Warshall to be a polynomial-time algorithm for all-pais longest paths?

13. Show that there is no absolute approximation algorithm for the max clique problem.

The following review problems are for your benefit. They include test-type problems, homework-type problems, and some more difficult exercises.

Exam 1. Review Problems:

0. 3-3, 4-1, 4-4

1. Find tight Big Oh formulations of each of the following (that is, remove the low order terms):
a) $n^2 + n(\log^4 n^2) + 1000$
b) $n^{\frac{3}{2}} + n \log n$
c) $\log n + \log^* n + \frac{1}{n}$

2. Which of a) and b) is asymptotically larger:
a) $n \log^2 n^2$
b) $n^2 \log n$

3. Find tight Big Omega formulation for:
$\log \log n + 100 + \frac{1}{n}$

4. Solve the following sum exactly:

$$\sum_{i=0}^{n} 2i + 1$$

5. Prove that

$$\sum_{k=1}^{n} \frac{2 \log k}{k} = O(\log^2 n)$$

(Hint: use solution to known sum to make it easier)

6a. Solve the following recurrence
T(1)=2
T(n)=10T(n/4) + 5n-7

6b. Show the following is $O(2^n)$. Use induction to show $T(n) \leq c2^n$, for some $c \geq 0$.
T(1)=1.

T(n)= $2^{\frac{n}{4}}$ T(n/2) + n

Exam 2. Review Problems:

1. 6.2-4
2. Let G=(V, E) be an undirected graph. A dominating set of G is a subset V' of V such that each v in V is adjacent to at least one u in V'. An independent set of G is a subset V" of V such that each v in V" is adjacent to no other v' in V". Prove that for any G there exists an independent set which is also a dominating set.
3. 23.1-5, 23.2-5
4. Prove Dijkstra's algorithm is correct (that is it finds a shortest path to each vertex), by using induction on the size of V1.
5. 26.2-2
6. 22.2-6
7. Let G be a graph with edge set E in which each vertex is colored either red or blue. Give a $O(E)$ time algorithm that finds a path from u to v that passes through the minimum number of blue vertices.
8. Give an $O(V + E)$ algorithm to find all cut-vertices in an undirected graph
9. Hard Problems: 22.2-7, 23.2-8
10. Let G be a weighted graph. Find a second shortest path from vertex u to vertex v (note that 2nd shortest path may possibly have same weight as shortest path).
11. 9.3-8
12. 9-1a, b
13. 9.1-1
14. 9.2-4
15. 23-3a