Name:

Fall 2017 Analysis of Algorithms

Comprehensive Examination

Score

1. (10 pts) Show that $\sum_{0 \le i < j < n} 1 = \binom{n}{2}$. Note, this sum can also be written as $\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$.

Score

2. (10 pts) Let A be an $n \times n$ matrix. Use summation notation to compute the big-O time complexity of the code:

```
int count [n];
for (int k = 0; k < n; k++) { count[k] = 0; }
for (int i = 0; i < n-1; i++) {
  for (int j = i+1; j < n; j++) {
    if (A[i][j] > 0) { count[j] = count[j] + 1; }
  }
}
```

Score

3. (10 pts) Assume n is a power of 2, that is, $n = 2^p$ for some natural number $p > 0$. Solve the recurrence

$$T(n) = 3T(n/2) + n, \qquad T(1) = 0$$

Score

4. (10 pts) The recursive cutRod algorithm below takes an array of prices p[] and a rod length n. It determines how to cut the rod to maximize the total value of the pieces.

```
int cutRod(int p[], int n) {
  if (n == 0) { return 0; }
  q = −∞;
  for (int k = 1; k <= n; k++) { q = max{q, p[k] + cutRod(p, n − k)};}
  return q
}
```

(a) What recurrence equation describes the time complexity $T(n)$ of cutRod?

(b) Show that $T(n) = 2^n$ satisfies the recurrence from part 4a.

Score

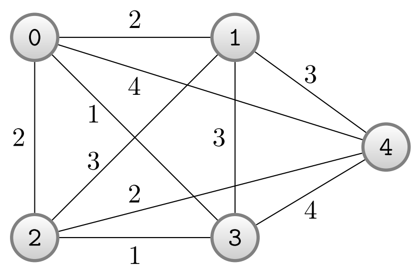
5. (10 pts) Here is a dynamic programming algorithm for the rod cutting problem. What is its time complexity?

```
int cutRod(int p[], n) {
  int r[n];
  r[0] = 0;
  for (int j=1; j <= n; j++) {
    q = −∞;
    for(int i=1; i <= j; i++) { q = max{q, p[i] + r[j − 1]}; }
    r[j] = q;
  }
  return r[n];
}
```

Score

6. (10 pts) Given a complete weighted graph, the traveling salesman problem (TSP) asks: What is the shortest possible route that visits each city exactly once and returns to the origin city?

   Starting from city 0, construct a route for the graph below and determine its cost.



Score

7. (10 pts) What heuristic was used to determine the route in problem 6?

Score

8. (10 pts) If the heuristic was developed into an algorithm, explain what its time complexity would be. Would the algorithm always produce an optimal route?

Score

9. (20 pts) Answer `True` or `False` for the following sentences, or explain that the answer is unknown.

   (a) All NP-hard problems are NP-complete.

   (b) The set of NP-complete problems is a subset of NP.

   (c) NP-complete problems cannot have polynomial algorithms.

   (d) In order to prove a problem X to be NP-complete one needs to develop a polynomial transformation from a known NP-complete problem to X.

   (e) 2-SAT is an NP-hard problem.

   (f) A tree with n nodes has $n-1$ edges.

   (g) There exists a polynomial-time algorithm for finding the maximum spanning tree in a undirected and weighted graph.

   (h) QuickSort has time complexity $O(n \lg n)$ when executed on a sorted array of size n where the pivot is always the head of the array.

   (i) There is a non-deterministic polynomial time algorithm that decides the halting problem.

   (j) This sentence is `False`.