New CS 473: Algorithms, Spring 2015
Homework 2

Due Tuesday, February 10, 2015 at 5pm

---

1. The ***maximum k-cut problem*** asks, given a graph G with edge weights and an integer k as input, to compute a partition of the vertices of G into k disjoint subsets $S_1, S_2, \ldots, S_k$ such that the sum of the weights of the edges that cross the partition (that is, have endpoints in different subsets) is as large as possible.

   (a) Describe an efficient $(1-1/k)$-approximation algorithm for this problem.

   (b) Now suppose we wish to minimize the sum of the weights of edges that do *not* cross the partition. What approximation ratio does your algorithm from part (a) achieve for the new problem? Justify your answer.

2. In the ***bin packing*** problem, we are given a set of n items, each with weight between 0 and 1, and we are asked to load the items into as few bins as possible, such that the total weight in each bin is at most 1. It's not hard to show that this problem is NP-Hard; this question asks you to analyze a few common approximation algorithms. In each case, the input is an array $W[1..n]$ of weights, and the output is the number of bins used.

```
NextFit(W[1..n]):
  bins ← 0
  Total[0] ← ∞
  for i ← 1 to n
    if Total[bins] + W[i] > 1
      bins ← bins + 1
      Total[bins] ← W[i]
    else
      Total[bins] ← Total[bins] + W[i]
  return bins
```

```
FirstFit(W[1..n]):
  bins ← 0
  for i ← 1 to n
    j ← 1; found ← False
    while j ≤ bins and ¬found
      if Total[j] + W[i] ≤ 1
        Total[j] ← Total[j] + W[i]
        found ← True
      j ← j + 1
    if ¬found
      bins ← bins + 1
      Total[bins] = W[i]
  return bins
```

   (a) Prove that NextFit uses at most twice the optimal number of bins.

   (b) Prove that FirstFit uses at most twice the optimal number of bins.

   *(c) ***[Extra Credit]*** Prove that if the weight array W is initially sorted in decreasing order, then FirstFit uses at most $(4 \cdot OPT + 1)/3$ bins, where OPT is the optimal number of bins. The following facts may be useful (but you need to prove them if your proof uses them):

   - In the packing computed by FirstFit, every item with weight more than $1/3$ is placed in one of the first OPT bins.
   - FirstFit places at most $OPT - 1$ items outside the first OPT bins.

3. Consider the following greedy algorithm for the metric traveling salesman problem: Start at an arbitrary vertex, and then repeatedly travel to the closest unvisited vertex, until every vertex has been visited.

   (a) Prove that the approximation ratio for this algorithm is $O(\log n)$, where n is the number of vertices. *[Hint: Argue that the kth least expensive edge in the tour output by the greedy algorithm has weight at most* $\mathrm{OPT}/(n-k+1)$*; try* $k=1$ *and* $k=2$ *first.]*

   *(b) ***[Extra Credit]*** Prove that the approximation ratio for this algorithm is $\Omega(\log n)$. That is, describe an infinite family of weighted graphs such that the greedy algorithm returns a Hamiltonian cycle whose weight is $\Omega(\log n)$ times the weight of the optimal TSP tour.

New CS 473 Spring 2015 — Homework 2 Problem 1

Name:	NetID:
Name:	NetID:
Name:	NetID:

(a) Describe an efficient $(1-1/k)$-approximation algorithm for the maximum k-cut problem.

(b) Now suppose we wish to minimize the sum of the weights of edges that do *not* cross the partition. What approximation ratio does your algorithm from part!(a) achieve for this new problem? Justify your answer.

New CS 473 Spring 2015 — Homework 2 Problem 2

Name:	NetID:
Name:	NetID:
Name:	NetID:

(a) Prove that NextFit uses at most twice the optimal number of bins.

(b) Prove that FirstFit uses at most twice the optimal number of bins.

(c) ***[Extra Credit]*** Prove that if the weights are initially sorted in decreasing order, then FirstFit uses at most $(4 \cdot OPT + 1)/3$ bins, where OPT is the optimal number of bins.

New CS 473 Spring 2015 — Homework 2 Problem 3

Name:	NetID:
Name:	NetID:
Name:	NetID:

1. Prove that the greedy algorithm for TSP has an approximation ration of $O(\log n)$.

*2. ***[Extra Credit]*** Prove that the approximation ratio for this algorithm is $\Omega(\log n)$.