

# CS270 Problem Set 2 – Spring 2005

Due: Tuesday, April 5

**Rules:** You can work in groups and consult papers and books. But you should write your paper by yourself, and you should give credit in it where credit is due, sources and colleagues. Please also write *briefly*, neatly, and correctly. Hard copy submissions are strongly preferred. If you submit late, write the date submitted at the top of your paper and bring it to (slip it under the door of) Soda 475.

1. **[10 pts] Project.** Write a 300-word executive summary of your project: problem definition, background, previous work, approach, desired results, and progress made so far.
2. **[12 pts] Linear Programming.** The max flow problem can be generalized in many directions. In each case below, show how to solve the more general problem by either (1) reducing it to the original max flow problem when possible, or (2) reducing it to linear programming in the remaining cases.
  - (a) There are many sources and many sinks, and we wish to maximize the total flow from all sources to all sinks.
  - (b) Each edge  $(v, w)$  has not only a capacity, but also a lower bound  $\ell(v, w)$  on the flow it must carry.
  - (c) Not only edges, but also nodes have capacities: upper bounds on the flow each node can carry.
  - (d) The outgoing flow from each node  $v$  is not the same as the incoming flow, but is smaller by a factor of  $(1 - \varepsilon_v)$ , where  $\varepsilon_v$  is the given loss coefficient associated with node  $v$ .
  - (e) Each edge has a cost per unit flow associated with it, and we must find among all flows of maximum value the one that minimizes the total cost.
  - (f) We must find the maximum flow each of whose paths use 5 or fewer edges (roughly, high bandwidth and low latency).
3. **[4 pts] Divide and Conquer.** Show how to find the decimal representation of an  $n$ -bit binary integer in time  $O(n^{1.6\dots})$ . (Hint: in class, we used Divide and Conquer to multiply two  $n$ -bit integers in the same amount of time.)
4. **[12 pts] Dynamic Programming.** Do any 3 of the following.
  - (a) Let  $X_1, \dots, X_n$  be independent random variables for which

$$X_i = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise.} \end{cases}$$

Let  $S = \sum_{i=1}^n X_i$ . Given  $s$ , show how to compute  $\Pr[S = s]$  as quickly as you can. How much time and space does your algorithm use? Your running time may be polynomial in  $n$  and  $s$ .

- (b) Describe an algorithm, as efficient as you can, that given a set of  $n$  points on the plane and a number  $k$ , finds the least “bumpy” left-to-right path that uses exactly  $k$  points. The *bumpiness* of a path is the sum of the absolute values of the angle changes on the path. **A left-to-right path is a sequence of points with increasing  $x$  coordinates.**
- (c) Describe an algorithm that, given a graph with edge weights and costs and two nodes  $s$  and  $t$ , finds the lowest-cost path from  $s$  to  $t$  of weight at most  $W$ . Your running time may be linear in  $W$ , and you can assume that every edge has weight at least 1, if you like.

- (d) You are asked to intersperse  $n$  *pop* operations into the sequence  $push(1), \dots, push(n)$  pushing integers onto a stack so that the sequence of integers popped is exactly a given permutation  $\pi(1), \dots, \pi(n)$  of  $1, \dots, n$ . Give a polynomial-time algorithm for determining whether this is possible.
5. [10 pts] **Coding. Definitions:** We deal only with binary codes here. A rate  $r$ ,  $n$ -bit code is an injective function  $f : \{0, 1\}^{rn} \rightarrow \{0, 1\}^n$ . Its relative distance is  $d/n$ , where  $d$  is the minimum Hamming distance between any two codewords.
- (a) Recall that a random linear code with rate  $r < C(\delta)$  has minimum **relative** distance  $\delta$  with high probability. What is the maximum number of errors one can tolerate with such a code? (*easy*)
- (b) How many rate  $r$ ,  $n$ -bit codes are there? How many linear rate  $r$ ,  $n$ -bit codes are there? (*easy*)
- (c) **Cancelled due to last minute bugs...**
6. [12 pts] **Integer Programming.** An integer program is a linear program in which the solutions are required to be integers. Integer programming is NP-hard, so any problem in NP can be reduced to it. Show how to write the following problems as integer programs.
- (a) MAX3SAT: Given a 3CNF formula, find an assignment which maximizes the number of satisfied clauses.
- (b) TSP: Given a weighted, undirected graph, find a minimum-weight tour of the graph — a cycle which visits all nodes exactly once.
- (c) VERTEX COVER: Given an undirected graph  $G = (V, E)$  and an integer  $k$ , select a set  $S \subseteq V$  of  $k$  nodes which maximizes the number of edges “touching”  $S$ , i.e. the number of edges  $(v, w)$  for which  $v \in S$  or  $w \in S$ .
- (d) 3D MATCHING: Given three disjoint sets  $W, X$ , and  $Y$  of  $n$  items, and a set  $M$  such that each element of  $M$  is a triple  $(w, x, y) \in W \times X \times Y$ , find a *maximum matching*: a largest subset of  $S$  of  $M$  such that each  $w \in W$ , each  $x \in X$ , and each  $y \in Y$  appears at most once among all triples in  $S$ . For example, if  $n = 3$  and  $M = \{(x_1, y_1, z_1), (x_1, y_1, z_2), (x_2, y_3, z_1), (x_3, y_2, z_3)\}$ , then the last three triples in  $M$  form a maximum matching (in this case we know it is maximum since no matching can have size greater than  $n$ ).
- (e) 2-PROCESSOR SCHEDULING: Given a list of integer job lengths  $\ell_1, \dots, \ell_n$ , find a *schedule* of the jobs on two processors with minimum *makespan*. A schedule assigns each job either to processor 0 or processor 1, and the makespan of a schedule is the total length of jobs on the most-loaded of the two processors.
- (f) DEGREE CONSTRAINED SPANNING TREE: Find a spanning tree of a given graph such that the maximum degree of a vertex in the tree is minimized.