Exercise 3 - Solutions

November 30, 2003

Question 1

Pick an interval I. "Break the circle" at the starting point of I (we can throw all intervals that intersect I). Now use the "Activity-selector" algorithm to find a set of non-intersecting intervals. The set we get is the largest such set containing I. To find the best set we can repeat the process for each of the intervals.

Question 2

Sort the columns of the matrix in non decreasing order of their lengths. Denote the columns in this order $v_1, \ldots, v_N$.

Algorithm:

- $B = \Phi$
- for i=1 to N do
  - if $B \cup \{v_i\}$ is linearly independent then
    - $B \leftarrow B \cup \{v_i\}$
- return B

The algorithm works since the columns of the matrix form a linear matroid. Its running time is $O(Nlog(N) + N \cdot T(n))$, where $T(n)$ is the running time of an algorithm that checks if a set of vectors is linearly independent.

Question 3

Define a matrix A of size $n \times (k+1)$. For every $v \in V(G)$

$$A(v,0) = \begin{cases} 1 & v = v_0 \\ 0 & v \neq v_0 \end{cases} \quad (1)$$

and for every $i > 0$

$$A(v,i) = \begin{cases} 1 & \textit{there is a path from } v_0 \textit{ to } v \textit{ labeled } < \sigma_1, \ldots, \sigma_k > \\ 0 & \textit{otherwise} \end{cases} \quad (2)$$

A can be defined recursively for every $v, i > 0$ like this:

$$A(v,i) = \bigvee_{u \to v} (A(u, i-1) = 1 \wedge \sigma(u,v) = \sigma_i)$$

We can find all the entries of A using dynamic programming, and the final answer will be 1 if there is a v such that $A(v,k) = 1$.

The running time is $O(n^2 k)$ since we have an $n \times (k+1)$ matrix to compute, and when computing one entry we have to look at no more than n other entries.

Question 4

Denote by g the greedy algorithm's choice. $g_1 \geq g_2 \geq \cdots \geq g_k$, and $\Sigma_{j=1}^{k} g_j = n$. Assume that the greedy algorithm is not optimal, therefore there exists an optimal solution o, $o_1 \geq o_2 \geq \cdots \geq o_m$, such that $\Sigma_{j=1}^{m} o_j = n$ and $m < k$. Let $i < k$ denote the minimal index such that $g_i \neq o_i$. Since g is the solution of the greedy algorithm, it must be that $g_i > o_i$. Furthermore, $g_i < \Sigma_{j=i}^{k} g_j = \Sigma_{j=i}^{m} o_j$.

(a)

All the weights of o_j for $j \geq i$ are less than g_i. but their sum is greater than g_i, and this means that o is not optimal - since it can easily be checked that for each of the cases $g_i \in \{25, 10, 5\}$ it must have a subsequence whose sum is exactly g_i, and it can be replaced by a single coconut of weight g_i. The case $g_i = 1$ cannot happen, since $g_i > o_i$, and $o_i \geq 1$.

(b)

It can be proved by induction on p that any sequence $o_i, o_{i+1}, \ldots, o_m$ such that $o_j < c^p$ and $\Sigma_{j=i}^{m} > c^p$ contains a subsequence whose sum is exactly c^p which can be replaced by a single coconut of weight c^p contradicting the optimality of o.

(c)

Let the weights of the coconuts be $\{\lceil \frac{n}{2} \rceil + 1, \lfloor \frac{n}{2} \rfloor, 1\}$. The greedy algorithm would choose $\lceil \frac{n}{2} \rceil + 1$, and then ones until it reaches n, while the optimal solution is choosing twice $\lfloor \frac{n}{2} \rfloor$ (and then maybe another 1 if n is odd).