EECS 477. HOMEWORK 6 SOLUTIONS.

1. Decision problems(20pts)

A. State the decision version of the knapsack problem: given n items of known weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fits into the knapsack. The objects cannot be broken.

Solution The decision version would sound like this: given n items of known weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$ and a knapsack of capacity W, *and given a positive integer* W, decide whether there is a subset of items that fits into the knapsack and whose value is greater or equal to K. The objects still cannot be broken.

B. Outline a polynomial-time algorithm that verifies whether or not a proposed solution (certificate) solves the knapsack decision problem.

Solution The algorithm is given a subset S of objects. It only does two things

A: Check that $\sum_{i \in S} w_i \leq W$

B: Check that $\sum_{i \in S} v_i \geq K$

If both of them are true then the certificate is the solution.

2. Paths(25pts)

Do problem 12.17(a) from the book

Solution Suppose that an algorithm is proposed that claims to decide on the existence of two-paths by checking less than $n(n-2)/2$ questions. Each question marks two cells in the symmetric adjacency matrix as seen, and let our daemon reply "no edge" to all those queries. Then after that, there is still at least on row where there are two unseen cells – daemon can put them either way on or off thus making the algorithm give wrong answer (having two different edges outgoing from a vertex is the same as having a two-path).

3. List scheduling

A.(25pts) Consider the following problem: we have n jobs, $J_1, \ldots, J_n$, and m identical machines, $M_1, \ldots, M_m$. Each job J_j must be processed without interruption for a time $p_j > 0$ on one of the m machines, each of which can process at most one job at a time. We need to find the schedule that minimizes the total makespan, that is, the time by which all jobs complete their processing.

Each schedule is specified by the sequences of jobs assigned to every machine in order they are processed. For instance, given three machines and five tasks with processing times

$$p_1 = 1, p_2 = 2, p_3 = 4, p_4 = 3, p_5 = 3.$$

Then one possible schedule will be specified as $[[J_2, J_3], [J_4], [J_1, J_5]]$. We can visualize this schedule with the following figure (the makespan is equal to 6 in this case):

The following **approximate** algorithm is proposed for this problem:

> the jobs are given in some particular order. Whenever a machine becomes available, the next job on the list is assigned to begin processing on that machine. If there is a tie, a machine with the smaller index takes the job.

Here is what you need to do: Show the schedule obtained by running the above algorithm on the instance with three machines ($m = 3$), and seven tasks $J_1, \ldots, J_7$ with processing times given as

$$p_1 = 5, p_2 = 2, p_3 = 4, p_4 = 3, p_5 = 6, p_6 = 1, p_7 = 12.$$

Illustrate it graphically.

Now run the same algorithm on the sequence of jobs in inverse order, and make an illustration.

What is the ordering of the jobs for which the above algorithm gives the worst answer (that is, the longest makespan)?

What is the schedule that minimizes the makespan?

See if your answers satisfy the statement of the part B of this problem.

Solution

Here is the schedule visualization:

```
 t:12345678901234567
M1:111116
M2:22444777777777777
M3:3333555555

A = 17 in this case
```

Inverse order schedule

```
 t:12345678901234567
M1:777777777777
M2:6444333311111
M3:55555522

A = 13 in this case
```

Now, the proof below actually implies that the worst possible time of list scheduling is within 5/3 of the optimal, that gives 18.333 which

means the worst we can get cannot be greater than 18. Here is a schedule that does it:

```
 t:12345678901234567
M1:555555777777777777
M2:11111226
M3:3333444

A = 18 in this case
the order of jobs indices is 5,1,3,4,2,7,6
```

Now the best solution cannot be better than the sum of processing times divided by the number of machines (that is $33/3 = 11$), or better than the longest job processing time (that is 12). Hence we cannot hope to get makespan better than 12 time units. Here is a schedule that achieves it:

```
 t:12345678901234567
M1:777777777777
M2:11111223333
M3:5555554446

makespan = 12 in this case which is optimal
```

as we can see, the results below are confirmed.

B.(extra 20 points) Prove that the above approximate algorithm always produces a solution whose makespan is at most twice the optimal one. That is, if the optimal makespan is T, the makespan obtained by the above algorithm is less than $2T$.

Solution Denote the makespan of the approximate solution by A. We need to show that $A \leq T$.

There are two observations: optimal makespan cannot be smaller than any individual processing time

$$T \geq p_k$$

for any k, and optimal makespan cannot be smaller than the one when all the machines are busy all the time till the end (the rectangle is fully packed), that is

$$T \geq \frac{1}{m} \sum_j p_j.$$

Another observation is that the *starting* time of the job k that finishes last cannot be later than the $(1/m) \sum_{j \neq k} p_j$ because otherwise would mean that at least one machine was without a job for some time before we started processing k. The following derivation then works:

$$A \leq p_k + (1/m) \sum_{j \neq k} p_j = (1 - 1/m) p_k + (1/m) \sum_j p_j \leq (1 - 1/m) T + T < 2T.$$

That proves it.

4. Graph problems (30 pts)

For the graph above,

A. Find all the hamiltonian cycles.

   The only hamiltonian cycle is ABDCE.

B. Find its chromaticity number (that is the minimal number of colors required to color the vertices of the graph so that no edge has the ends of the same color).

   The chromaticity number is 3. It cannot be less because there are triangles in the graph.

C. A *clique* is a set of nodes such that there is an edge in the graph between any two nodes in the clique. Find the size of the largest clique in the above graph.

   Again the answer is 3.