

# ICS 260 – Fall 2001 – Final Exam

Name: **Answer Key**

Student ID:

1:

2:

3:

4:

5:

6:

Total:

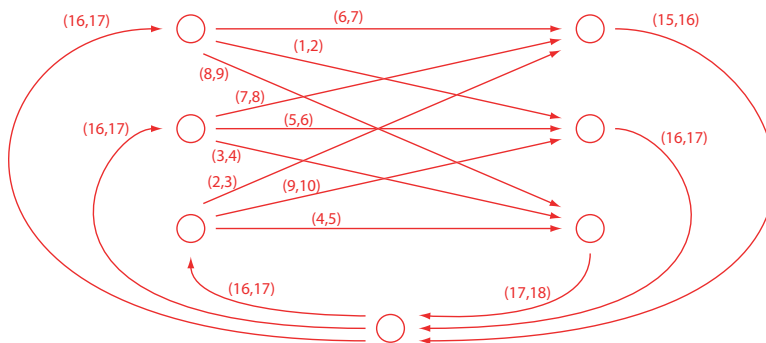
1. Matrix rounding. (25 points)

Suppose we are given the following  $3 \times 3$  matrix as input (shown also with the sums of each row and of each column).

63	14	84	161
71	53	38	162
21	94	49	164
155	161	171	

We wish to output a rounded form of the matrix, where each entry is rounded up or down to the nearest multiple of ten, as are the row and column sums. The rounded sums should still be an accurate sum of the row and column, however, so e.g. it would not be allowed to round every entry in the first row of the matrix downwards because that would cause the sum of the rounded row to differ from the rounded sum of the row.

Draw a graph, with upper and lower bounds on the flow amounts for each edge, such that a feasible circulation in this graph corresponds to a solution to the matrix rounding problem. You do not need to solve the feasible circulation problem in your graph, and you do not need to find a rounded form of the matrix.



It would also have been ok to split the bottom vertex into two vertices. Grading criteria:

- 25 points for a correct answer. I didn't take anything off (but perhaps I should have) for making all upper and lower bounds be multiples of ten – that would work, but then you need a circulation with flow amounts that are multiples of ten and not just integers.
- 20 points for forgetting to make the graph directed (except in one case where the capacities were set up in such a way that it worked anyway as an undirected graph).
- 20 points for a solution that looked like the one above but omitted the loop back from the right side to the left side (this should be a feasible circulation problem, not a flow problem).
- 20 points for a solution that forgot to label some of the edges.
- 10 points for a solution with the right shape of graph but no bounds on the edges.
- 5 points for a solution that started with some correct ideas but then went very wrong.

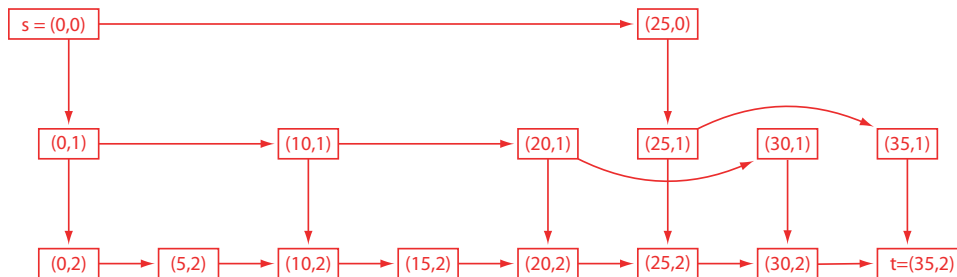
2. Change-making problem. (30 points)

Suppose you are making change for  $n$  cents with three types of coins, worth 5, 10, and 25 cents (e.g. the US nickel, dime, and quarter). For example, there are six different ways of making change for  $n = 35$ :  $25+10$ ,  $25+5+5$ ,  $10+10+10+5$ ,  $10+10+5+5+5$ ,  $10+5+5+5+5+5$ , and  $5+5+5+5+5+5+5$ . You may assume that  $n$  is a multiple of five.

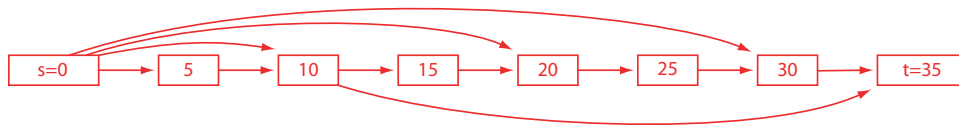
Describe a general technique for transforming such a change-making problem into a directed acyclic graph, with two of the graph vertices labeled as  $s$  and  $t$ , in such a way that the paths from  $s$  to  $t$  in the graph correspond to the different ways of making change for  $n$  cents. Also draw the graph resulting from using your technique on the example  $n = 35$ .

Your technique must produce graphs of size  $O(n)$  for the input  $n$ . For full credit, no two paths in the graph should represent the same way of making change (so your example with  $n = 35$  should have exactly six paths from  $s$  to  $t$ ). No credit will be given to techniques that work by listing all methods of making change and creating a separate path for each such method.

My intended solution: if you have  $k$  coins  $C[0], \dots, C[k-1]$ , draw a graph in which each vertex is labeled by a pair of numbers  $(x, i)$  where  $0 \leq x \leq n$  and  $i < k$ . Draw edges from each vertex  $(x, i)$  to  $(x, i+1)$  and  $(x, i)$  to  $(x + C[i], i)$ . Let  $s = (0, 0)$  and  $t = (n, k-1)$ . Then the paths from  $s$  to a vertex  $(x, i)$  represent the different ways of making change for  $x$  cents using only coins of types  $C[0], \dots, C[i]$ . There are  $kn$  vertices and  $(2k-1)n$  edges; since  $k = 3 = O(1)$ , both of these quantities are  $O(n)$ . Example for  $n = 35$ , with  $C = (25, 10, 5)$ , showing only the vertices reachable from  $s$ :



One student came up with an alternative solution, which produces a somewhat smaller graph, but more strongly depends on the assumption that  $k = 3$ : Label each vertex with a number  $x$  instead of a pair, and let  $s$  be the vertex labeled 0 and  $t$  be the vertex labeled  $n$ . Connect each vertex  $x$  by an edge to vertex  $x + C[0]$ , connect vertex 0 by an edge to each vertex  $x$  for which  $x$  is a multiple of  $C[1]$ , and connect each vertex  $y$  to  $n$  if  $n - y$  is a multiple of  $C[2]$ . Here is an example for  $n = 35$  with the coins reordered to  $C = (5, 10, 25)$ , again only showing the reachable vertices:



Grading criteria:

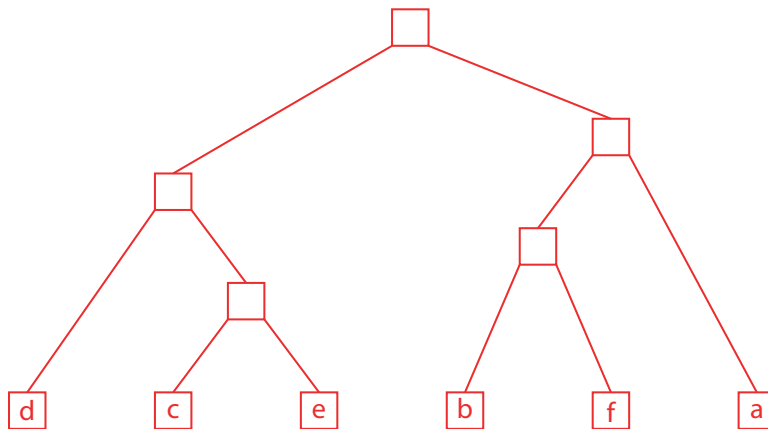
- 30 points for a correct answer.
- 25 points for an answer with the right idea but minor mistakes.
- 20 points for a solution with one row of vertices (like the second solution above) but connecting  $x$  to each  $x + C[i]$ , so that there are multiple paths per change solution.
- 15 points for a set of vertices in  $k$  rows but with the wrong edge connections.

3. Huffman coding. (20 points)

Suppose we are given the following input to the greedy Huffman code construction algorithm:

symbol	frequency
a	12
b	5
c	3
d	9
e	4
f	6

Describe the Huffman code that would be constructed by the algorithm from this input. You may either show it as a tree, or write the sequence of binary digits used to encode each symbol.



Grading criteria:

- 20 points for a correct tree.
- 10 points for an incorrect tree with some evidence of understanding the correct procedure (e.g. one student forgot to include one of the symbols).
- 5 points for other incorrect trees.

#### 4. PC-board drilling. (25 points)

Suppose you are managing a PC-board manufacturing factory. One of the steps each PC-board must undergo is drilling holes for components: most components are surface-mounted, but a few are placed in holes and then soldered. You must design a program that determines the sequence in which the holes are drilled in each board.

You may assume the following: It takes 1/2 second to drill a hole, and one second per inch to move the drill to each successive hole. Each board needs roughly 20 holes, and there are typically 200 identical boards to be drilled in a job lot. You are willing to dedicate a powerful PC to the drill sequencing task, and it can work on finding the sequence for one job lot while the previous job lot is being drilled. The drilling step is the bottleneck for your factory, so the faster you can perform it the more business you can handle, but you can not afford to buy more than one drill.

Would it be more appropriate to solve this drill sequencing problem using Christofides' heuristic, or by branch-and-bound? Explain your answer.

The intended answer is that both procedures can solve the problem (which is a variation of the traveling salesman problem) but that branch-and-bound would be better because the answer quality is much more important than running time: branch-and-bound will eventually give the optimal answer, while the answer from Christofides might be as much as 50% away from optimal. The size of the problem (20 points) is so small that you will very likely finish the branch-and-bound within the hours of time available per problem, and even if it didn't complete it will keep track of the best answer found so far which will likely be pretty good.

In fact modern branch-and-bound systems such as <http://www.math.princeton.edu/tsp/concorde.html> can solve TSP problems much more quickly than needed for this application – I didn't discuss details such as this in lecture, so I set up the question with very extreme parameters so that even if you lost a few orders of magnitude in your estimates you'd likely come to the same conclusion.

Some students suggested a further refinement: run Christofides first, then use that for the bound in the branch-and-bound procedure. That way you are guaranteed to be at least as good as Christofides, but likely will get a much better solution.

Grading criteria:

- 25 points for ideas like the ones discussed above.
- 20 points for a serious attempt to compare the running times, plugging in the numbers given, and concluding that branch-and-bound's worst case of  $O(n!)$  is just too slow even for 20 points.
- 15 points for saying that branch-and-bound is better because it can more flexibly handle changes to the model – this is true, but Christofides matches the stated problem too.
- 10 points for saying branch and bound is worse because it's exponential, without examining whether this makes sense for the specific parameters stated here.
- 5 points for recognizing that this is a TSP problem which can be approximated by Christofides, without addressing why that would be a better choice than branch-and-bound.
- 5 points for saying that branch-and-bound works, without addressing why that would be a better choice than Christofides.
- 5 points for saying that branch-and-bound isn't suited for this kind of problem.
- 0 points for a choice without an explanation.
- 0 points for a description of an alternative solution technique without any explanation of why it would be better than the two stated in the question.
- 0 points for describing the two algorithms.

5. Complexity theory. (10 points)

For each of the following sentences, state whether the sentence is known to be true, known to be false, or whether its truth value is still unknown.

(a) If a problem is in P, it must also be in NP.

**TRUE.**

(b) If a problem is in NP, it must also be in P.

**UNKNOWN.**

(c) If a problem is NP-complete, it must also be in NP.

**TRUE.**

(d) If a problem is NP-complete, it must not be in P.

**UNKNOWN.**

(e) If a problem is not in P, it must be NP-complete.

**FALSE.**

**Grading criteria: 2 points for each correct answer.**

6. NP-completeness (40 points). If you get at least 30 points on this question, you will be guaranteed at least a B+ for your overall course grade.

In a certain town, there are many clubs, and every adult belongs to at least one club. The townspeople would like to simplify their social life by disbanding as many clubs as possible, but they want to make sure that afterwards everyone will still belong to at least one club.

Prove that the Redundant Clubs problem is NP-complete. You may make use of the known NP-completeness of the Maximum Independent Set, Set Cover, or Traveling Salesman Problems (all described below).

PROBLEM NAME: Redundant Clubs

INPUT: List of people; list of clubs; list of members of each club; number  $K$ .

OUTPUT: Yes if there exists a set of  $K$  clubs such that, after disbanding all clubs in this set, each person still belongs to at least one club. No otherwise.

PROBLEM NAME: Maximum Independent Set

INPUT: Undirected graph  $G$  and number  $K$ .

OUTPUT: Yes if there exists a set of  $K$  vertices in  $G$  such that no pair of vertices in the set is connected by an edge. No otherwise.

PROBLEM NAME: Set Cover

INPUT: List  $L$  of elements, family  $F$  of subsets of elements, number  $K$ .

OUTPUT: Yes if there exist some  $K$  subsets in the family the union of which is all of  $L$ . No otherwise.

PROBLEM NAME: Traveling Salesman Problem

INPUT: Matrix of distances between cities, number  $K$ .

OUTPUT: Yes if there is a tour that visits each city once and has total length at most  $K$ , no otherwise.

First, we must show that Redundant Clubs is in NP, but this is easy: if we are given a set of  $K$  clubs, it is straightforward to check in polynomial time whether each person is a member of another club outside this set.

Next, we reduce from a known NP-complete problem, Set Cover. We translate inputs of Set Cover to inputs of Redundant Clubs, so we need to specify how each Redundant Clubs input element is formed from the Set Cover instance. We use the Set Cover's elements as our translated list of people, and make a list of clubs, one for each member of the Set Cover family. The members of each club are just the elements of the corresponding family. To finish specifying the Redundant Clubs input, we need to say what  $K$  is: we let  $K = F - K_{SC}$  where  $F$  is the number of families in the Set Cover instance and  $K_{SC}$  is the value  $K$  from the set cover instance. This translation can clearly be done in polynomial time (it just involves copying some lists and a single subtraction).

Finally, we need to show that the translation preserves truth values. If we have a yes-instance of Set Cover, that is, an instance with a cover consisting of  $K_{SC}$  subsets, the other  $K$  subsets form a solution to the translated Redundant Clubs problem, because each person belongs to a club in the cover. Conversely, if we have  $K$  redundant clubs, the remaining  $K_{SC}$  clubs form a cover. So the answer to the Set Cover instance is yes if and only if the answer to the translated Redundant Clubs instance is yes.

Grading criteria:

- 10 points for checking that Redundant Clubs is in NP.
- 15 points for doing the translation in the correct direction: from an input to a known NP-complete problem, to an input for the Redundant Clubs problem rather than vice versa.
- 15 points for a correct translation. The most common mistake here was to use  $K = K_{SC}$  instead of  $K = F - K_{SC}$ .