Homework 6 Solution

Spring 2018

1

Suppose we want to make change for n cents and the only denominations allowed are 1, 10, and 25 cents (infinite amount).

a

Find an example such that the greedy algorithm does not find the minimum number of coins required to make change for n cents (give a concrete counterexample).

Using the greedy algorithm to make change for $n = 30$ cents does not yield the smallest amount of coins. The greedy algorithm suggests that we take one quarter and five pennies, however, the minimum amount is actually three dimes.

b

Is it possible to design a coin system (a set of possible denominations) such that a greedy algorithm yields an optimal solution? If so, find such an example with at least 3 denominations.

The coin set $\{25, 10, 5, 1\}$ always yields an optimal solution. Other working coin systems include sets that all numbers are a multiple of all smaller numbers in the same set, such as $\{1, 2, 4, 8\}$, $\{1, 2, 6, 12\}$ etc.

2

What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21

Can you generate your answer to find the optimal code when the frequencies are the first n Fibonacci numbers?

An optimal Huffman code for the first 8 Fibonacci numbers are as follows:

h: 0

g: 10

f: 110

e: 1110

d: 11110

c: 111110

b: 1111110

a: 1111111

Let 1^i here indicate there are i amount of repeating 1 bits. An optimal code when the frequencies are the first n Fibonacci numbers follow this pattern:

nth character: 0

ith character: $1^{n-i}0$

1st character: 1^{n-1}

3

For the given graph below, show the three representations of the graph according to how we discussed in class.

Adjacency Matrix:

	a	b	c	d	e	f	g	h	i	j
a	0	1	0	0	1	1	0	0	0	0
b	1	0	1	0	0	0	1	0	0	0
c	0	1	0	1	0	0	0	1	0	0
d	0	0	1	0	1	0	0	0	1	0
e	1	0	0	1	0	0	0	0	0	1
f	1	0	0	0	0	0	0	1	1	0
g	0	1	0	0	0	0	0	0	1	1
h	0	0	1	0	0	1	0	0	0	1
i	0	0	0	1	0	1	1	0	0	0
j	0	0	0	0	1	0	1	1	0	0

Adjacency List:

A $\rightarrow$ B $\rightarrow$ E $\rightarrow$ F

B $\rightarrow$ A $\rightarrow$ C $\rightarrow$ G

C $\rightarrow$ B $\rightarrow$ D $\rightarrow$ H

D $\rightarrow$ C $\rightarrow$ E $\rightarrow$ I

E $\rightarrow$ A $\rightarrow$ D $\rightarrow$ J

F $\rightarrow$ A $\rightarrow$ H $\rightarrow$ I

G $\rightarrow$ B $\rightarrow$ I $\rightarrow$ J

H $\rightarrow$ C $\rightarrow$ F $\rightarrow$ J

I $\rightarrow$ D $\rightarrow$ F $\rightarrow$ G

J $\rightarrow$ E $\rightarrow$ G $\rightarrow$ H

Incidence Matrix:

	ab	bc	cd	de	ea	af	bg	ch	di	ej	gj	jh	hf	fi	ig
a	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
b	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
c	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
d	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0
e	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
f	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
g	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
h	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
i	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1
j	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Would you consider this graph sparse? If so, why? Justify your answer.

This graph is sparse. There are 10 vertices with only 15 edges. A dense matrix would have much closer to the maximum amount of edges, which for this graph would be: $\frac{10*9}{2} = 45$ edges.

4

Suppose we have n students in a class and a list of r student pairs to indicate the two people have dated before. A date only involves two people here. Give an $\mathcal{O}(n + r)$-time algorithm that determines whether it is possible to group students into two groups such that no one has ever dated someone from the same group.

This is a test of whether or not a graph is bipartite:

Start from any vertex v_1 and color it red.

Color neighbors of v_1 blue.

Proceed by coloring all neighbors of an already colored vertex the opposite color.

If a contradiction occurs, then it is not bipartite, and it is not possible to make this grouping.

Otherwise, this will complete in $\mathcal{O}(n + r)$.

5

One can use DFS to determine whether or not a given undirected graph $G = (V, E)$ contains a cycle. Does the DFS here run in $\mathcal{O}(|V| + |E|)$ time or just $\mathcal{O}(|V|)$, independent of $|E|$ time?

A graph is acyclic if a DFS yields no back edges. In an acyclic forest, there are at most $|V| - 1$ edges. Therefore, $|E| \leq |V| - 1$ and if we encounter $|V|$ edges, then we know to stop, since we have encountered a cycle. Thus, this runs in $\mathcal{O}(|V|)$.