COMP 3711H – Fall 2016
Tutorial 6

1. (CLRS–16.2-4) Professor Midas drives an automobile from Newark to Reno along Interstate 80. His car's gas tank, when full, holds enough gas to travel m miles, and his map gives the distance between gas stations on his route. The professor wishes to make as few gas stops as possible along the way. Give an efficient method by which Professor Midas can determine at which gas stations he should stop and prove that your algorithm yields an optimal solution.

   The simple greedy algorithm is optimal. It is to drive to the furthest possible city that one can reach with the current gas in the car and then fill up the tank and then continue.

   Suppose that the cities are at locations $0 = x_0 < x_1 < \ldots < x_n$.

   Let GREEDY be the greedy solution which we will denote by G. We will prove optimality of greedy by induction on n. Let O be any optimal solution and assume that Greedy is optimal on all problems on set size $< n$ *(for the basis this is obviously true on sets of size 1 and 2). We may also assume that, whenever O adds gas, O fills the gas tank completely (since this can not make the solution worse). We use* $|O|$ *and* $|G|$ *to denote the numbers of stops each solution makes.*

   Now consider the input on n points. Let g_1 *be the first stop that Greedy makes and* o_1 *be the first stop that OPT makes. By the definition of Greedy,* $o_1 \leq g_1$. *Write*

$$\begin{aligned} G &= g_1, g_2, \ldots, g_k \\ O &= o_1, o_2, \ldots, o_{k'} \end{aligned}$$

   By definition, $k' \leq k$, *where the* g_i *and* o_i *are the stops the algorithms make. Let* i *be the first index for which* $g_i \neq o_i$.

   Now let $t = \max_i o_i \leq g_1$. *From the observations above we know that* $t \geq 1$.

   Set

$$O' = g_1, o_{t+1}, o_{t+2}, \ldots, o_{k'}.$$

   Since $g_1 \geq o_t$ *this is a legal tour. Thus* $t = 1$, *otherwise O was not optimal. So*

$$O' = g_1, o_2, o_3, \ldots, o_{k'}.$$

   Now note that $o_2, o_3, \ldots, o_{k'}$ *must be an optimal stopping pattern for the problem* $x_{g_1}, x_{g_1+1}, \ldots, x_n$ *because otherwise we could replace it in O with a smaller set of gas fills, getting a smaller optimal solution (which is imposisble).*

   From the induction hypothesis we know that $g_2, \ldots, g_k$ *is an optimal stopping pattern for the problem* $x_{g_1}, x_{g_1+1}, \ldots, x_n$.

   Thus $k = k'$ *and greedy is optimal for our original set.*

2. Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

   (a) Describe a greedy algorithm to make change consisting of quarters (25 cents) dimes (10), nickels (5), and pennies (1). Prove that your algorithm yields an optimal solution.

(b) Suppose that the available coins are in denominations that are powers of c. i.e. the denominations are $c^0, c^1, ..., c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

(c) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.

(a) *Set* $c_q = \lfloor n/25 \rfloor$,
This is the largest number of quarters that can be used to make change for n *cents.*
Set $n_q = n - 25c_q$.
This is the amount remaining after using c_q *quarters*
Set $c_d = \lfloor n_q/10 \rfloor$ *(largest number of dimes that can be used)*
Set $n_d = n_q - 10c_d$ *(amount remaining)*
Set $c_n = \lfloor n_d/5 \rfloor$
Set $c_p = n_p = n_d - 5c_n$.

Solution uses c_q *quarters,* c_d *dimes,* c_n *nickles and* c_p *pennies.*

Proof of optimality: Assume Greedy solution G *is not optimal.*

Let O *be an optimal solution using* o_q *quarters,* o_d *dimes,* o_n *nickels and* o_p *pennies.*
First note that if $o_p \geq 5$ *we can replace every 5 pennies with one nickle, reducing the number of coins used, so we can assume* $o_p < 5$.

If $o_d \geq 3$ *replace every three dimes with 1 quarter and 1 nickle without increasing the number of coins. So, we can assume* $o_d \leq 2$.
If $o_n \geq 2$ *replace every 2 nickels with one dime, reducing the number of coins used, so we can assume* $o_n \leq 1$.

Now suppose that $10o_d + 5o_n + o_p \geq 25$. *The only way that this can happen is if* $o_d = 2$ *and* $o_n = 1$. *In this case we can replace the two dimes and one nickle with one quarter, reducing the number of coins used, contradicting optimality of* O. *So this is impossible.*
This means that $10o_d + 5o_n + o_p < 25$. *Since*

$$n = 25o_q + 10o_d + 5o_n + o_p$$

we have just shown that $c_q = \lfloor n/25 \rfloor = o_q$.
After greedy takes off the $c_q = o_q$ *quarters what remains is*

$$n' = n - 25o_q = 10o_d + 5o_n + o_p.$$

From the facts that $o_n \leq 1$ *and* $o_p \leq 4$ *we see that* $5o_n + o_p < 10$ *so Greedy chooses* $c_d = o_d$ *dimes and then* $c_n o_n$ *nickles and then* $c_p = o_p$ *pennies, so we are done.*

(b) Recall that there is a unique way to write n *in base* c, *i.e.,*

$$n = \sum_i a_i c^i \quad \text{such that} \quad \forall i,\ 0 \leq a_i < c.$$

First consider the greedy solutions. Suppose it chooses g_i *coins of type* c^i. *If, for some* i, $g_i \geq c$ *then the greedy solution could have chosen one more coin of denomination*

c^{i+1}. So, for all i, $g_i < c$, and by the uniqueness of such a representation, for all i, $g_i = a_i$.

Now let o_i denote the number of c^i coins used in an optimal solution for making change of n cents. Note that, $\forall i$, $o_i < c$; if this was not true then we could replace the o_i coins of size c_i with one coin of size c^{i+1} and $(o_i - c)$ coins of size c_i, reducing the number of coins used, contradicting optimality of O.

This, for all i, $o_i < c$ which means that, for all i, $o_i = a_i$.

Since, for all i, $g_i = a_i = o_i$, greedy IS optimal.

(c) Let $1, 4, 6$ be the set of coin denominations.

Suppose we make change for $n = 8$ cents.

The greedy solution uses one 6 cent coin and two 1 cent coins, i.e. it uses 3 coins.

However, the optimal solution would only use two 4 cents coins.

3. Given an undirected graph $G = (V, E)$, its *complement*, $\overline{G}$, is the graph (V, E') such that for all $u \neq v$, $\{u, v\} \in E'$ if and only if $\{u, v\} \notin E$. Prove that either G or $\overline{G}$ is connected.

   Let $\bar{E}$ be the edges in $\overline{G}$. If G is not connected thene there exists some pair u, v) that have no path in G connecting them.

   - *$\{u, v\} \in \bar{E}$ so u, v are connected in $\overline{G}$.*
   - *For all other vertices w, at least one of $\{u, w\}$ and $\{u, w\}$ must be in $\bar{E}$; otherwise both of them are in E which would have connected u, v.*
   - *Then every other vertex w is connected to u (and v) in $\overline{G}$ since such a w must contain an edge to at least one of u, v and is connected to the other via the edge $\{u, v\}$.*
   - *Then every pair w, w' are connected in $\overline{G}$ since they are both connected to u.*

4. An (undirected) graph $G = (V, E)$ is *bipartate* if there exists some $S \subset V$ such that, for every edge $\{u, v\} \in E$, either (i) $u \in S$, $v \in V - S$ or (ii) $v \in S$, $u \in V - S$.

   Let $G = (V, E)$ be a connected graph. Design an $O(n + e)$ algorithm that checks whether G is bipartite. *Hint: Run BFS.*

   Run BFS. Recall that $d[v]$ will store the shortest distance from the root v.

   Set S to be the set of all vertices with $d[v]$ even.

   G will be bipartite if and only if all edges (u, v) in the graph satisfy that the parity of $d[v]$ and $d[u]$ are not the same, i.e., $d[v]$ is odd and $d[u]$ is even or vice versa.

   If the condition is true, it is easy to see that G is, by definition bipartite. Just set S to be the set of all even vertices.

   If the graph is bipartite, let $S, V - S$ be the bipartite split and assume without loss of generality that $w \in S$. Then the length of every *path from w to other nodes in S is even and the length of every path to nodes in $V - S$ is odd. In particular the lengths of the* shortest *paths to nodes in S are even and to those in $V - S$ are odd. So, the partity of the endpoints of all edges in G must be different.*

   How can you modify your algorithm so that it also works for unconnected graphs?

   No modification is needed because a graph is bipartite if and only if each of its connected components is bipartite.