20/12/2011
E 10.1

Exercise 4

Consider the following problems:

1) Vertex-cover (VC)

Given an undirected graph $G=(V,E)$ and an integer $k \geq 2$, is there a subset C of V with $|C| \leq k$ such that C covers all edges of G (i.e., for each edge $\{v_i, v_j\} \in E$ with $v_i \neq v_j$, $\{v_i, v_j\} \cap C \neq \emptyset$).

2) Independent-set (IS)

Given an undirected graph $G=(V,E)$ and an integer $k \geq 2$, is there a subset C of V with $|C| \geq k$ such that for all $v_i, v_j \in C$ with $v_i \neq v_j$, $\{v_i, v_j\} \notin E$

3) Clique

Given an undirected graph $G=(V,E)$ and an integer $k \geq 2$, is there a subset C of V with $|C| \geq k$ such that for all $v_i, v_j \in C$ with $v_i \neq v_j$, $\{v_i, v_j\} \in E$.

Show that VC, IS, and Clique can be reduced to each other in polynomial time.

N.B. In the definitions of VC, IS, and Clique we have ignored self-loops (since we required $v_i \neq v_j$).

IS $\leq_{poly}$ Clique

Given an instance (G, k) of IS, we construct an instance (G', k') of Clique as follows:

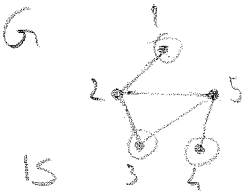
$k' = k$

Let $G = (V, E)$.

Then $G' = (V, E')$, where $E' = V \times V \setminus E$

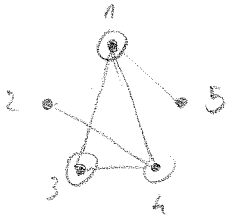
(i.e., the edges of E' are obtained by connecting all pairs of nodes that are not connected in E.)

E.g. G



IS

G'



Clique

The reduction works because the maximum independent set of G is precisely the maximum clique in the complement graph of G.

VC $\leq_{poly}$ IS

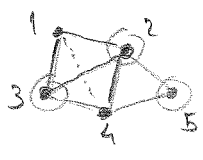
Given an instance (G, k) of VC, we construct an instance (G', k') of IS as follows:

$k' = |V| - k$

$G' = G$

The reduction works because the vertexes in a vertex-cover C cover all edges of G. Hence the set $V \setminus C$ must have no edges between its elements, and is thus an independent set

E.g.



The marked nodes $\{2,3,5\}$ are a VC of size 3. Hence, there cannot be an edge $\{1,4\}$

Clique $\leq_{poly}$ VC

Given an instance (G, k) of Clique, we construct an instance (G', k') of VC as follows:

$k' = |V| - k$

Let $G = (V, E)$.

Then $G' = (V, E')$, where $E' = V \times V \setminus E$

Exercises on space complexity

Exercise 1: Let A be an algorithm of space complexity $s(n)$.

Show that there is an algorithm A' such that

- $\mathcal{L}(A) = \mathcal{L}(A')$
- A' has space complexity $s'(n) = O(s(n) + \log n)$
- A' does not scan the input-tape beyond the boundaries of the input

Proof: we proceed in two steps

1) We prove that on input x, there is an algorithm A_x such that
   - $\mathcal{L}(A_x) = \mathcal{L}(A)$
   - A_x does not scan the input-tape beyond location $2^{O(s(n) + \log_2 n)}$ from the input

This proof is analogous to the one that we did in class to show that a poly-space bounded (N)TM is equivalent to one that has running time $t(n) \leq c^{q(n)}$ with $q(n) = O(s(n))$ (where $s(n)$ is a polynomial space bound)

We showed $q(n) = 2 \cdot s(n) + d$, where $c = |\Gamma| + |Q|$

In our case:

$$\Rightarrow c^{q(n)} = 2^{\log_2 c \cdot q(n)} = 2^{O(s(n))}$$

- we have also the position on the input tape that contributes to the configuration:

$$\Rightarrow n \cdot 2^{O(s(n))} = 2^{\log_2 n} \cdot 2^{O(s(n))} = 2^{O(s(n) + \log_2 n)}$$

different configurations at most

Note: a TM with running time $t(n) \leq 2^{O(s(n) + \log n)}$ can scan at most $2^{O(s(n) + \log n)}$ cells of the input tape

2) We modify the algorithm A_1 in such a way that it does not move beyond the input.

The resulting algorithm A' works as follows:

- whenever A_1 would move right past the end of the input, A' instead:
  - does not move past the end of the input, but maintains a counter on the work tape
  - whenever A_1 moves right, the counter is incremented
  - " " left, " " decremented

  In this way, A' can keep track of the position of the input head of A_1

  Whenever A_1 moves back again over the input symbol, A' does not update the counter (leaving it to 0)

- A' operates similarly whenever A_1 moves left past the beginning of the input

How much space does the counter use:

Since A_1 does not scan the input tape beyond $s_I(n) = 2^{O(s(n)+\log n)}$ the counter takes $\log_2 s_I(n) = O(s(n) + \log n)$

Hence, the total space used by A' is

$$s(n) + O(s(n) + \log 2) = O(s(n) + \log n)$$

E 10.6

Exercise 2: Let A be an algorithm of space complexity s.

Show that there is an algorithm A' such that

- A' computes the same function as A, i.e. $A'(w) = A(w)\ \forall w \in \{0,1\}^*$
- A' has space complexity $s'(n) = s(n) + O(\log \ell(n))$

  where $\ell(n) = \max_{w \in \{0,1\}^n} |A(w)|$ is the size of the maximum output for input x of length n
- A' never rewrites on the same location of its output tape

Proof:

A' proceeds in successive iterations, each time simulating the whole computation of A:

in the i-th iteration, A' outputs the i-th bit of $A(x)$

When emulating A, in its i-th iteration A' proceeds as follows:

- it does not directly (re)write on the output tape
- instead, it maintains on the work tape:
  - the counter i of the next output bit that will be written
  - a counter c of the bit that A is currently writing
  - the value of the bit written by A in position i
- when A would write an output bit, A' operates depending on the values of i and c:
  - if $i \neq c$, then A' does not output anything
  - if $i = c$, then A' stores the written bit on its worktape
- At the end of its simulation, A' outputs the stored bit to the i-th position of the output tape

How much space $s'(n)$ does A' use on the working tape for inputs of length n?

- $s(n)$ cells, since it performs the computation of A
- the space for the counters i, and c_0.
  - i, c_1 and c_0 have to count positions on the output tape, and hence will use $\log_2 \ell(n)$ bits each.

We get that $s'(n) = s(n) + O(\log_2 \ell(n))$