Prof. Friedrich Eisenbrand
Martin Niemeier

Due Date: April 28, 2009
Discussions: April 21, 2009

Introduction to Discrete Optimization

Spring 2009

Solutions 8

Exercise 1
What is the smallest number n such that an algorithm A with running time $1000000 \cdot n^2$ is faster than an algorithm B with a running time of 2^n?

Solution
Let $f(n) := 1000000 \cdot n^2$ and $g(n) := 2^n$.

We have $f(29) = 841,000,000$ and $g(29) = 536,870,912$, but $f(30) = 900,000,000$ and $g(30) = 1,073,741,824$. Thus $n = 30$ is the smallest number such that A is faster than B.

Exercise 2
Decide which ones of the following statements are true and give a short explanation:

1. $4n^3 + 3n^2 - n - 100 = O\left(n^3\right)$
2. $n = O\left(n^7\right)$
3. $n^3 - 100n^2 = \Omega(n^4)$
4. $2^n = O\left(n^2\right)$
5. $\sqrt{n} = O(n)$
6. $\log(n) = O\left(\sqrt{n}\right)$

Solution

1. Let $T(n) := 4n^3 + 3n^2 - n - 100$. If $n \geq 1$ we have

$$4n^3 + 3n^2 - n - 100 \leq 7n^3.$$

   Thus if we set $n_0 := 1$ and $c := 7$, then $T(n) \leq cn^3$ for each $n \in \mathbb{R}_{\geq 0}$, $n \geq n_0$ and thus $T(n) = O\left(n^3\right)$.

2. Let $T(n) := n$. If $n \geq 1$ we have

$$n \leq n^7.$$

   Thus if we set $n_0 := 1$ and $c := 1$, then $T(n) \leq c \cdot n^7$ for each $n \in \mathbb{R}_{\geq 0}$, $n \geq n_0$ and thus $T(n) = O\left(n^7\right)$.

3. Let $T(n) := n^3 - 100n^2$. Assume that $T(n) = \Omega(n^4)$. Thus there are constants $c \in \mathbb{R}_{\geq 0}$ and $n_0 \in \mathbb{R}_{\geq 0}$ such that

$$n^3 - 100n^2 \geq c \cdot n^4 \tag{1}$$

for each $n \geq n_0$.

Equation (1) implies

$$0 \leq -c \cdot n^4 + n^3 - 100n^2 = -n^2 \cdot (c \cdot n^2 + n - 100) \tag{2}$$

for each $n \geq n_0$.

Let $n' := \max\{n_0, 100\}$. Then

$$-\underbrace{(n')^2}_{>0} \cdot (\underbrace{c \cdot (n')^2}_{>0} + \underbrace{n' - 100}_{\geq 0}) < 0$$

in contradiction to (2). Thus $T(n) = \Omega(n^4)$ does NOT hold.

4. Let $T(n) := 2^n$. Assume that $T(n) = O\left(n^2\right)$. Thus there are constants $c \in \mathbb{R}_{\geq 0}$ and $n_0 \in \mathbb{R}_{\geq 0}$ such that

$$2^n \leq c \cdot n^2 = 2^{\log(c \cdot n^2)} = 2^{\log(c) + 2\log(n)}$$

for each $n \geq n_0$. This implies $n \leq \log(c) + 2\log(n)$ and thus $f(n) := -n + \log(c) + 2\log(n) \geq 0$ for each $n \geq n_0$.

Observe that $f'(n) = -1 + \frac{2}{\ln(2)} \frac{1}{n} \longrightarrow_{n \to \infty} -1$. Thus there is some $n' \geq n_0$ such that $f(n') < 0$ wich is a contradition.

Thus $2^n \neq O\left(n^2\right)$.

5. Let $T(n) := \sqrt{n} = n^{\frac{1}{2}}$. If $n \geq 1$ we have

$$n^{\frac{1}{2}} \leq n.$$

Thus if we set $n_0 := 1$ and $c := 1$, then $T(n) \leq c \cdot n$ for each $n \in \mathbb{R}_{\geq 0}$, $n \geq n_0$ and thus $T(n) = O(n)$.

6. Let $T(n) := \log n$ and $f(n) := \sqrt{n} - \log n$.

Observe that for each $n \geq 0$ we have $\log(n) \leq \sqrt{n}$ if and only if $f(n) \geq 0$.

Then $f'(n) = n^{-\frac{1}{2}} - n^{-1} \cdot \frac{1}{\ln(a)} = n^{-\frac{1}{2}} \cdot (1 - n^{-\frac{1}{2}} \cdot \frac{1}{\ln(2)}) \longrightarrow_{n \to \infty} 1$. Thus for some n_0 large enough we have

$f(n_0) \geq 0$ and $f(n) \geq 0$ for all $n \geq n_0$. Thus $log(n) = O\left(\sqrt{n}\right)$.

Exercise 3

Consider the following algorithm. The input is a sequence of n integers $a[1], \ldots, a[n]$.

Require:
1: **for** $i \leftarrow 1$ to n **do**
2: **for** $j \leftarrow n$ downto $i+1$ **do**
3: **if** a[j] < a[j-1] **then**
4: exchange $a[j] \leftrightarrow a[j-1]$
5: **end if**
6: **end for**
7: **end for**
8: Output $a[1], \ldots, a[n]$.

Give an asymptotic upper bound on the running time of the algorithm (with explanations). What is the output of the algorithm?

Solution
This algorithm is the so called BUBBLE-SORT algorithm. It sorts the input sequence and outputs it in non-decreasing order.

Each execution of the body of the inner loop (line 3-5) takes constant time, since at most two operations (one comparison and one exchange) are carried out.

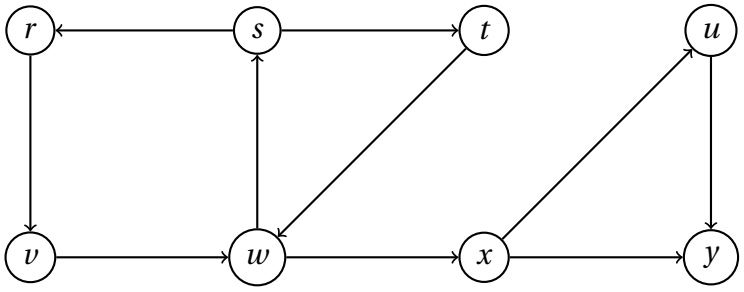
The outer loop is executed n times, and in the i-th invokation, the inner look is executed $n-i$ times.

Thus in total, the body of the inner loop is executed

$$\sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n(n+1)}{2} = \frac{n^2}{2} - \frac{n}{2} = O\left(n^2\right).$$

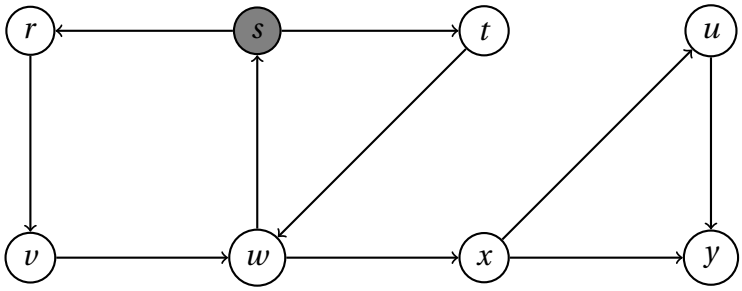
Thus an asymptotic upper bound on the running time of the algorithm is $O\left(n^2\right)$.

Exercise 4
Consider the following graph:



Perform the breath first search algorithm on this graph starting in s. For each node v, give the values $\pi[v]$ and $d[v]$ at the end of the algorithm.

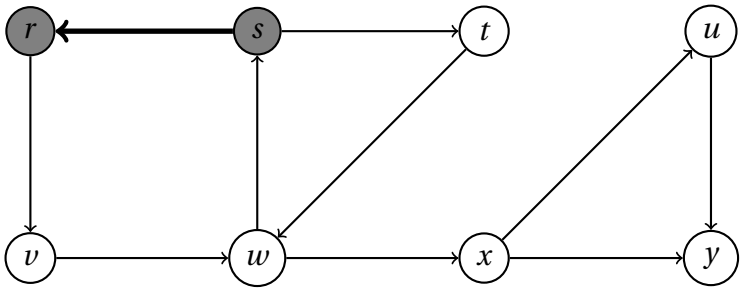
Solution
Initialization:



$d[r] = \infty,\ d[s] = 0,\ d[t] = \infty.\ d[u] = \infty, d[v] = \infty,\ d[w] = \infty,\ d[x] = \infty,\ d[y] = \infty$

$\pi[r] = 0,\ \pi[s] = 0,\ \pi[t] = 0,\ \pi[u] = 0,\ \pi[v] = 0,\ \pi[w] = 0,\ \pi[x] = 0,\ \pi[y] = 0$

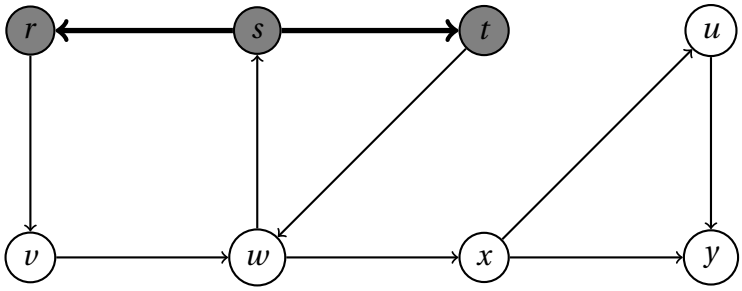
1. Iteration:



$d[r] = 1,\ d[s] = 0,\ d[t] = \infty.\ d[u] = \infty, d[v] = \infty,\ d[w] = \infty,\ d[x] = \infty,\ d[y] = \infty$

$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = 0,\ \pi[u] = 0,\ \pi[v] = 0,\ \pi[w] = 0,\ \pi[x] = 0,\ \pi[y] = 0$

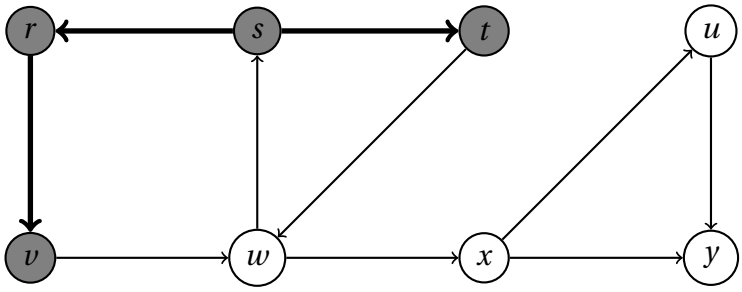
2. Iteration:



$d[r] = 1,\ d[s] = 0,\ d[t] = 1.\ d[u] = \infty, d[v] = \infty,\ d[w] = \infty,\ d[x] = \infty,\ d[y] = \infty$

$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = s,\ \pi[u] = 0,\ \pi[v] = 0,\ \pi[w] = 0,\ \pi[x] = 0,\ \pi[y] = 0$

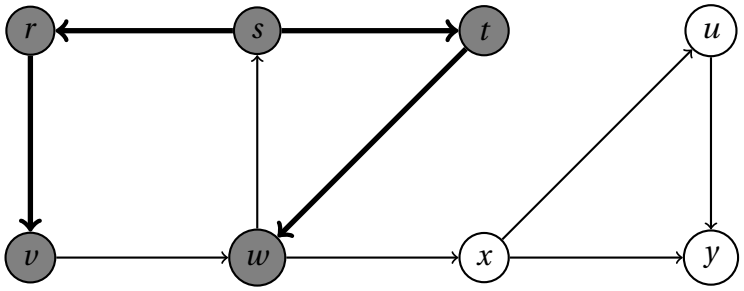
3. Iteration:



$$d[r] = 1,\ d[s] = 0,\ d[t] = 1.\ d[u] = \infty, d[v] = 2,\ d[w] = \infty,\ d[x] = \infty,\ d[y] = \infty$$

$$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = s,\ \pi[u] = 0,\ \pi[v] = r,\ \pi[w] = 0,\ \pi[x] = 0,\ \pi[y] = 0$$

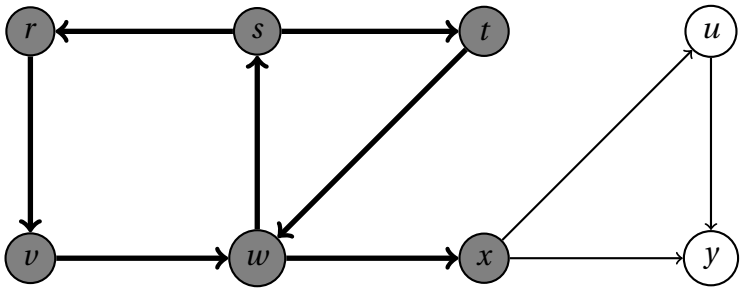
4. Iteration:



$$d[r] = 1,\ d[s] = 0,\ d[t] = 1.\ d[u] = \infty, d[v] = 2,\ d[w] = 2,\ d[x] = \infty,\ d[y] = \infty$$

$$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = s,\ \pi[u] = 0,\ \pi[v] = r,\ \pi[w] = t,\ \pi[x] = 0,\ \pi[y] = 0$$

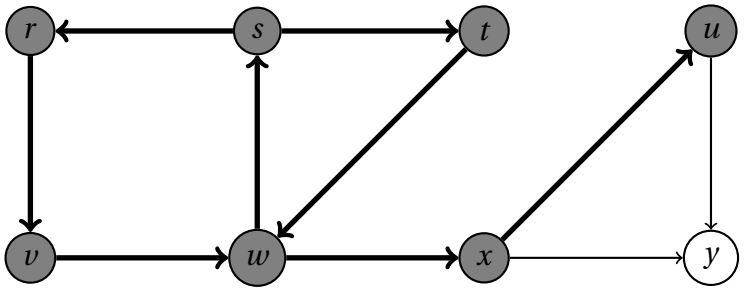
5. Iteration:



$$d[r] = 1,\ d[s] = 0,\ d[t] = 1.\ d[u] = \infty, d[v] = 2,\ d[w] = 2,\ d[x] = 3,\ d[y] = \infty$$

$$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = s,\ \pi[u] = 0,\ \pi[v] = r,\ \pi[w] = t,\ \pi[x] = w,\ \pi[y] = 0$$

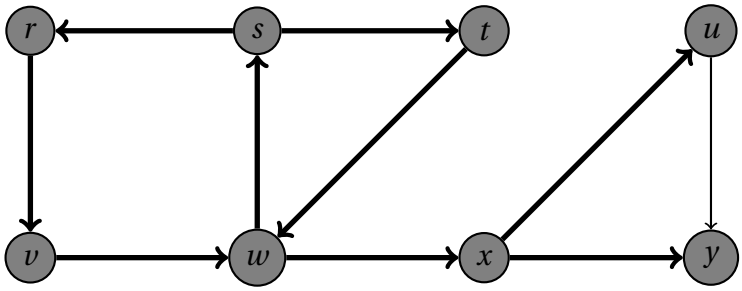
6. Iteration:



$$d[r] = 1,\ d[s] = 0,\ d[t] = 1.\ d[u] = 4, d[v] = 2,\ d[w] = 2,\ d[x] = 3,\ d[y] = \infty$$

$$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = s,\ \pi[u] = x,\ \pi[v] = r,\ \pi[w] = t,\ \pi[x] = w,\ \pi[y] = 0$$

7. Iteration:



$d[r] = 1,\ d[s] = 0,\ d[t] = 1.\ d[u] = 4, d[v] = 2,\ d[w] = 2,\ d[x] = 3,\ d[y] = 4$

$\pi[r] = s,\ \pi[s] = 0,\ \pi[t] = s,\ \pi[u] = x,\ \pi[v] = r,\ \pi[w] = t,\ \pi[x] = w,\ \pi[y] = x$

Exercise 5

There are two types of professional wrestlers: "good guys" and "bad guys". Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an $O(n+r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as good guys and the remainder as bad guys such that each rivalry is between a good guy and a bad guy.

If it is possible to perform such a designation, your algorithm should produce it.

Hint: Describe the rivalries as a graph and use an algorithm from the lecture to solve the problem.

Solution

Let V be the set of wrestlers. Let $A := \{(u,v) \ : \ u,v \in V, u \text{ and } v \text{ have a rivalry.}\}$. Consider the graph $G = (V,A)$. A set $U \subseteq V$ is called *independent*, if there are no arcs between two nodes of U, i.e. for all $(uv) \in A$ we have $u \notin U$ or $v \notin U$.

Note that a designation of the wrestlers as "good guys" and "bad guys" with no rivalries inside the two groups is equivalent to find a partition of the nodes of G into two sets V_1 and V_2, i.e. $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$, such that V_1 and V_2 are independent.

Now perform breadth first search starting in any node $v \in V$. If there are nodes left that have not been visited in the breadth first search, do another breadth first search starting at one of these nodes. Repeat this until all nodes have been visited.

Partition the nodes into two sets V_1 and V_2 defined as follows:

$$V_1 := \{v \in V \ : \ d[v] \text{ is uneven}\}.$$

$$V_2 := \{v \in V \ : \ d[v] \text{ is even}\}.$$

Assume that V_1 is not independent. Thus there are two nodes $u,v \in V_1$ such that $(u,v) \in A$. Let $s \in V$ be the starting node of the breadth first search. Since $d[u]$ and $d[v]$ are uneven, there is an $s-u$ path and an $s-v$ path of uneven length in d. By construction of V, for each arc $(v_1,v_2) \in A$ we have $(v_2,v_1) \in A$ as well. Thus there is an $v-s$-path of uneven length in G as well. Concatenation of the $s-u$-path, the arc (u,v) and the $v-s$-path gives a cycle C in G of uneven length. This shows that it is not possible to partition the nodes of G into two

independent sets: However one distributes the nodes of the cycle to the sets V_1 and V_2, at least one pair of "neighbors" will end up in V_1 or V_2.

If V_2 is not independent, we can argue analogously that it is not possible to partition the nodes of G into independent sets.

Thus, the algorithm described above yields a feasible designation if and only if it is possible to find it. Since the number of nodes in the graph is n and the number of arcs is $2r$, its running time is $O(n+r)$.