Fundamental Algorithms - Spring 2018
Homework 9

Daniel Rivera Ruiz
Department of Computer Science
New York University
drr342@nyu.edu

1. Let us define $GAP(a,b) = L - (l_a + \ldots + l_b + b - a)$ for a line that contains words with lengths l_a through l_b. Let us now suppose that we know the value k such that the words with lengths $l_k \ldots l_i$ will be in the last line of the text. The value of $FBAD[i]$ can therefore be computed as

$$FBAD[i] = FBAD[k-1] + P[GAP(k,i)]$$

   Where the value of $FBAD[k-1]$ is already known since $k \leq i$. However, we cannot know the value of k a priori, and therefore we need to consider all the possible values and take the one that minimizes the previous expression. If we consider the extreme case where all words have length one, the maximum number of words that can fit in one line are $\frac{L}{2}$ (where the remaining $\frac{L}{2}$ would be occupied by white spaces between the words). Finally the formula for $FBAD[i]$ is as follows:

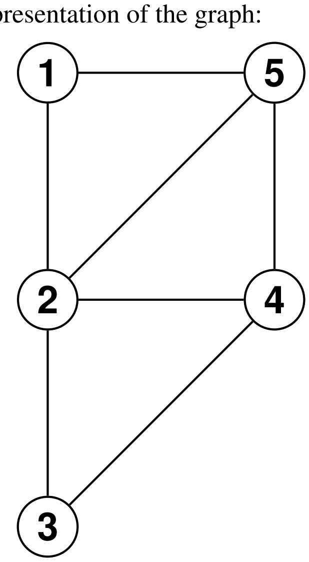
$$FBAD[i] = \min_k \{FBAD[k-1] + P[GAP(k,i)]\}, i - \frac{L}{2} \leq k \leq i$$

   The algorithm to calculate $FBAD[i]$ based on the previous formula is as follows:

```
FBAD[i] = ∞
for k = i-L/2 to i:
    TEMP = FBAD[k-1] + P[GAP(k, i)]
    if (FBAD[i] > TEMP):
        FBAD[i] = TEMP
    end if
end for
return FBAD[i]
```

   We observe that all the operations inside the `for` loop take constant time and there are $\frac{L}{2}$ iterations, therefore the time complexity of the algorithm is given by $T = O\left(\frac{L}{2}\right) = O(L)$.

2. The following image shows a representation of the graph:



The following table shows the partial results during the execution of BFS with $s = 3$:

Q	u	$V = adj[u]$	color$[V]$	$d[V]$	$\pi[V]$
$\{3\}$	3	$\{2, 4\}$	$\{w, w\}$	$\{1, 1\}$	$\{3, 3\}$
$\{2, 4\}$	2	$\{1, 5, 3, 4\}$	$\{w, w, b, g\}$	$\{2, 2, -, -\}$	$\{2, 2, -, -\}$
$\{4, 1, 5\}$	4	$\{2, 5, 3\}$	$\{b, g, b\}$	$\{-, -, -\}$	$\{-, -, -\}$
$\{1, 5\}$	1	$\{2, 5\}$	$\{b, g\}$	$\{-, -\}$	$\{-, -\}$
$\{5\}$	5	$\{4, 1, 2\}$	$\{b, b, b\}$	$\{-, -, -\}$	$\{-, -, -\}$

Finally, we get the values for d and π:

$$d[\{1, 2, 3, 4, 5\}] = \{2, 1, 0, 1, 2\}$$
$$\pi[\{1, 2, 3, 4, 5\}] = \{2, 3, -, 3, 2\}$$

3. The following table shows the partial results during the execution of BFS on the graph of figure A with $s = u$:

Q	u	$V = adj[u]$	color$[V]$	$d[V]$	$\pi[V]$
$\{u\}$	u	$\{t, x, y\}$	$\{w, w, w\}$	$\{1, 1, 1\}$	$\{u, u, u\}$
$\{t, x, y\}$	t	$\{u, w, x\}$	$\{b, w, g\}$	$\{-, 2, -\}$	$\{-, t, -\}$
$\{x, y, w\}$	x	$\{t, u, w, y\}$	$\{b, b, g, g\}$	$\{-, -, -, -\}$	$\{-, -, -, -\}$
$\{y, w\}$	y	$\{u, x\}$	$\{b, b\}$	$\{-, -\}$	$\{-, -\}$
$\{w\}$	w	$\{s, t, x\}$	$\{w, b, b\}$	$\{3, -, -\}$	$\{w, -, -\}$
$\{s\}$	s	$\{r, w\}$	$\{w, b\}$	$\{4, -\}$	$\{s, -\}$
$\{r\}$	r	$\{s, v\}$	$\{b, w\}$	$\{-, 5\}$	$\{-, r\}$
$\{v\}$	v	$\{r\}$	$\{b\}$	$\{-\}$	$\{-\}$

Finally, we get the values for d and π:

$$d[\{r, s, t, u, v, w, x, y\}] = \{4, 3, 1, 0, 5, 2, 1, 1\}$$
$$\pi[\{r, s, t, u, v, w, x, y\}] = \{s, w, u, -, r, t, u, u\}$$

4. Let $L = \{v_1, v_2, \ldots, v_n\}$ be the list with all the nodes (boxers) in the graph and consider the following algorithm:

```
initialize():
    for i = 1 to n:
        COLOR(v_i) ← WHITE;
        TYPE(v_i) ← NULL;
    end for

BFS(s):
    COLOR(s) ← GRAY
    Q.PUSH(s)
    while Q ≠ {∅}:
        u ← Q.POP()
        for all v ∈ adj(u):
            if TYPE(v) ≠ NULL and TYPE(v) ≠ TYPE(s):
                return FALSE
            end if
            if COLOR(v) = WHITE:
                COLOR(v) ← GRAY
                TYPE(v) ← TYPE(s)
                Q.PUSH(v)
            end if
        end for
        COLOR(u) ← BLACK
    end while
return TRUE
```

```
26  BFS-MASTER(L):
27      initialize()
28      TYPE(v_1) ← GOOD
29      BFS(v_1)
30      i ← 2
31      while TYPE(v_i) = GOOD
32          i++
33          if i > n
34              return FALSE
35          end if
36      end while
37      TYPE(v_i) ← BAD
38      flag = BFS(v_i)
39      if flag = FALSE
40          return FALSE
41      end if
42      for i = 1 to n:
43          if TYPE(v_i) = NULL
44              return FALSE
45          end if
46      end for
47      print({v | TYPE(v) = GOOD})
48      print({v | TYPE(v) = BAD})
49  return TRUE
```

The algorithm works as follows:

- On line 27 we initialize the fields `COLOR` and `TYPE` for all the nodes with `WHITE` and `NULL`, respectively (see lines 1 to 5).
- On line 28 we assign the first node v_1 to type `GOOD` and run `BFS` on it. After BFS finishes, all the nodes that are reachable from v_1 will also have type `GOOD` (see line 18).
- On lines 30 to 36 we find the first element that was not reached in the previous step. If no such element exists, i.e. all the elements are reachable from v_1 (the graph is fully connected), the algorithm terminates and returns `FALSE`.
- On line 38 we run `BFS` on the node v_i found in the previous step. Our version of `BFS` will return `TRUE` if it terminates successfully, and `FALSE` if at any point it encounters a node which had already been reached by the first call to `BFS(`v_1`)` (see lines 13 to 15).
- If `BFS(`v_i`)` returns `FALSE`, it means that there is at least one node that can be reached by both groups `GOOD` and `BAD`, and so the algorithm returns `FALSE` and terminates (lines 39 to 41).
- Finally, we run a loop over all nodes to see if they have all been reached. If at least one node was never reached, the algorithm returns `FALSE` and terminates (lines 42 to 46). Otherwise, the two sets of nodes are printed and the algorithm finishes successfully (lines 47 to 49).

To estimate the time complexity of the algorithm we notice that `initialize()` takes time $O(n)$, as do the `while` and `for` loops. Additionally, each call to `BFS` takes time $O(n + r)$. Since all the operations are executed sequentially, the overall time complexity for `BFS-MASTER` will be $O(n + r)$.

5. Table 1 shows the evolution of the DFS algorithm on the graph of figure B, starting at node r and considering all adjacent lists to be alphabetical. To build the table, we notice that the evolution of the stack corresponds to the changes of color in the nodes of the graph: whenever a node is added to the stack its color changes from white to gray, and whenever a node is removed its color changes from gray to black.

   Finally, we gather from the table the values for d and f:

$$d[\{r, q, s, t, u, v, w, x, y, z\}] = \{01, 04, 05, 11, 02, 06, 07, 12, 03, 13\}$$
$$f[\{r, q, s, t, u, v, w, x, y, z\}] = \{20, 17, 10, 16, 19, 09, 08, 15, 18, 14\}$$

6. Table 2 is similar to the one from the previous exercise, in this case for the graph of figure C starting at node m. Based on this table, `TOP-SORT` will output the vertices according to their finishing times $f[i]$ but in reverse order:

$$\texttt{TOP-SORT}(G_C) = \{p, n, o, s, m, r, y, v, x, w, z, u, q, t\}$$

7. (a) Since G is a DAG, we can build a linear version of it where all the nodes are in a straight line in the order defined by `TOP-SORT`. Assuming that the game starts at the first node, with n nodes in the line and all the edges going from left to right (because of `TOP-SORT`), there can be at most $n-1$ edges (in the case where there is an edge between all consecutive nodes) before reaching the last node in the line. Since each edge in the graph corresponds to a move in the game, it follows that the game must end after $n-1$ moves at the most.

   (b) The following algorithm returns `VALUE[z]` when `VALUE[w]` is given for all $w \in Adj[z]$:

```
FIND-WINNER[z]:
    for all w ∈ Adj[z]:
        if VALUE[w] = DOS:
            return UNO
        end if
    end for
return DOS
```

   - In the base case $Adj[z]$ is empty i.e., z is a leaf and therefore the `for` loop is never executed: we return `DOS` (as explained in the assignment).
   - If we find a node with value `DOS`, it means that Player 1 can move from z to that node and assure the victory. Therefore we return `UNO` and terminate.
   - If all the nodes in the set have value `UNO`, no matter what Player 1 does he will leave the game in a position where Player 2 can win, therefore we return `DOS`.

   (c) The following variation of the `DFS-VISIT` algorithm would find the winner of the game if it started at vertex v:

```
DFS-VISIT[v]:
    COLOR[v] ← GRAY
    for all w ∈ Adj[v]:
        if COLOR[w] = WHITE:
            DFS-VISIT[w]
        end if
    end for
    COLOR[v] ← BLACK
    VALUE[v] ← FIND-WINNER[v]
```

   The time complexity of the modified algorithm is $O(E)$ (the same as the original algorithm) because there is only one extra `for` loop (implicit in the `FIND-WINNER` routine). This loop executes at most E times and therefore the asymptotic analysis remains the same.

Table 1:

Time	Node n	$M = adj[n]$	color$[M]$	Stack	Interpretation
1	–	$\{r\}$	$\{w\}$	$\{r\}$	$d[r]$
2	r	$\{u, y\}$	$\{w, w\}$	$\{r, u\}$	$d[u]$
3	u	$\{y\}$	$\{w\}$	$\{r, u, y\}$	$d[y]$
4	y	$\{q\}$	$\{w\}$	$\{r, u, y, q\}$	$d[q]$
5	q	$\{s, t, w\}$	$\{w, w, w\}$	$\{r, u, y, q, s\}$	$d[s]$
6	s	$\{v\}$	$\{w\}$	$\{r, u, y, q, s, v\}$	$d[v]$
7	v	$\{w\}$	$\{w\}$	$\{r, u, y, q, s, v, w\}$	$d[w]$
8	w	$\{s\}$	$\{g\}$	$\{r, u, y, q, s, v\}$	$f[w]$
9	v	$\{w\}$	$\{b\}$	$\{r, u, y, q, s\}$	$f[v]$
10	s	$\{v\}$	$\{b\}$	$\{r, u, y, q\}$	$f[s]$
11	q	$\{s, t, w\}$	$\{b, w, b\}$	$\{r, u, y, q, t\}$	$d[t]$
12	t	$\{x, y\}$	$\{w, g\}$	$\{r, u, y, q, t, x\}$	$d[x]$
13	x	$\{z\}$	$\{w\}$	$\{r, u, y, q, t, x, z\}$	$d[z]$
14	z	$\{x\}$	$\{g\}$	$\{r, u, y, q, t, x\}$	$f[z]$
15	x	$\{z\}$	$\{b\}$	$\{r, u, y, q, t\}$	$f[x]$
16	t	$\{x, y\}$	$\{b, g\}$	$\{r, u, y, q\}$	$f[t]$
17	q	$\{s, t, w\}$	$\{b, b, b\}$	$\{r, u, y\}$	$f[q]$
18	y	$\{q\}$	$\{b\}$	$\{r, u\}$	$f[y]$
19	u	$\{y\}$	$\{b\}$	$\{r\}$	$f[u]$
20	r	$\{u, y\}$	$\{b, b\}$	$\{\}$	$f[r]$

Table 2:

Time	Node i	$J = adj[i]$	color$[J]$	Stack	Interpretation
1	–	$\{m\}$	$\{w\}$	$\{m\}$	$d[m]$
2	m	$\{q, r, x\}$	$\{w, w, w\}$	$\{m, q\}$	$d[q]$
3	q	$\{t\}$	$\{w\}$	$\{m, q, t\}$	$d[t]$
4	t	$\{\}$	$\{\}$	$\{m, q\}$	$f[t]$
5	q	$\{t\}$	$\{b\}$	$\{m\}$	$f[q]$
6	m	$\{q, r, x\}$	$\{b, w, w\}$	$\{m, r\}$	$d[r]$
7	r	$\{u, y\}$	$\{w, w\}$	$\{m, r, u\}$	$d[u]$
8	u	$\{t\}$	$\{b\}$	$\{m, r\}$	$f[u]$
9	r	$\{u, y\}$	$\{b, w\}$	$\{m, r, y\}$	$d[y]$
10	y	$\{v\}$	$\{w\}$	$\{m, r, y, v\}$	$d[v]$
11	v	$\{w\}$	$\{w\}$	$\{m, r, y, v, w\}$	$d[w]$
12	w	$\{z\}$	$\{w\}$	$\{m, r, y, v, w, z\}$	$d[z]$
13	z	$\{\}$	$\{\}$	$\{m, r, y, v, w\}$	$f[z]$
14	w	$\{z\}$	$\{b\}$	$\{m, r, y, v\}$	$f[w]$
15	v	$\{w, x\}$	$\{b, w\}$	$\{m, r, y, v, x\}$	$d[x]$
16	x	$\{\}$	$\{\}$	$\{m, r, y, v\}$	$f[x]$
17	v	$\{w, x\}$	$\{b, b\}$	$\{m, r, y\}$	$f[v]$
18	y	$\{v\}$	$\{b\}$	$\{m, r\}$	$f[y]$
19	r	$\{u, y\}$	$\{b, b\}$	$\{m\}$	$f[r]$
20	m	$\{q, r, x\}$	$\{b, b, b\}$	$\{\}$	$f[m]$
21	–	$\{n\}$	$\{w\}$	$\{n\}$	$d[n]$
22	n	$\{o, q, u\}$	$\{w, b, b\}$	$\{n, o\}$	$d[o]$
23	o	$\{r, s, v\}$	$\{b, w, b\}$	$\{n, o, s\}$	$d[s]$
24	s	$\{r\}$	$\{b\}$	$\{n, o\}$	$f[s]$
25	o	$\{r, s, v\}$	$\{b, b, b\}$	$\{n\}$	$f[o]$
26	n	$\{o, q, u\}$	$\{b, b, b\}$	$\{\}$	$f[n]$
27	–	$\{p\}$	$\{w\}$	$\{p\}$	$d[p]$
28	p	$\{o, s\}$	$\{b, b\}$	$\{\}$	$f[p]$