Algorithms – CMSC-27230

Walk to path

László Babai

Last updated 2024-02-08

Exercise Let $G = (V, E)$ be a digraph and $u, v \in V$. We describe an algorithm that, given a $u \to \cdots \to v$ walk, constructs a $u \to \cdots \to v$ path in linear time, i. e., time $O(n + k)$ where $n = |V|$ and k is the length of the walk. We assume $V = [n] = \{1, \ldots, n\}$.

The vertices are represented by "tokens," i. e., binary strings of length $\lceil \log_2(n + 1) \rceil$. Each token contributes one unit to the input size, and operations on tokens (addition/subtraction, comparison, copying, viewing the token as a link and following the link) costs one unit.

Solution The procedure moves along the walk W. Each time it reaches a vertex x, it jumps forward to the last occurrence of x along W and continues from there. To implement the algorithm in linear time, we need to store the last occurrence of each vertex; this can be done in a single pass along the walk.

Input: n in unary (number of vertices of the digraph)
. array $W = [w_0, \ldots, w_k]$ walk of length k where $w_0 = u, w_k = v$
(Note: the digraph itself is not part of the input.)

Output: array $P = [z_0, \ldots, z_\ell]$ path, where $z_0 = u, z_k = v$.

We say that for $0 \le i \le k$, the *position* i holds vertex w_i. It is a frequent mistake to fail to make this distinction; the essence of the problem is the fact that multiple positions can hold the same vertex. If $x = w_i$, we shall also say that x *occurs* at position i along the walk.

The idea of the algorithm is that we create an array of length n called "last" which, with each vertex, holds its last position. (If a vertex does not occur in W then its last position will be NIL.) With the help of this array we shall be able to skip forward from any vertex along the walk to its last occurrence.

Procedure Walk-to-path [input as listed above]

```
01   for j = 1 to n
02       last[j] := NIL              (: initializing the "last" array :)
03   end(for)
04   for i = 0 to k
05       last[w_i] := i              (: completing the "last" array :)
06   end(for)
07   ℓ := 0,    z_0 := x := w_0      (: initializing path :)
08   while x ≠ w_k
09       ℓ := ℓ + 1
10       x := w_{1+last[x]}          (: advance along path :)
11       z_ℓ := x                    (: appending next vertex to path :)
12   end(while)
13   for i = 0 to ℓ
14       P[i] := z_i                 (: listing path :)
15   return P
```

Analysis

CORRECTNESS

Notation 1 Let $U = \{w_0, \ldots, w_k\}$ denote the set of vertices of the walk W.

Observation 2 If $x \in U$ then $\text{last}[x] := \max\{i : x = w_i\}$.
If $x \in V \setminus U$ then last[x] =NIL.

Observation 3 If $x \in U$ then $x = w_{\text{last}[x]}$.

Proof. Immediate by Observation 2. □

Notation 4 $U^* = U \setminus \{w_k\} = \{w_0, \ldots, w_{k-1}\}$.

Notation 5 For $x \in U^*$ we write $N(x) = w_{1+\text{last}[x]}$.

Observation 6 If $x \in U$ and $x \neq w_k$ then $\text{last}[x] < k$.

Proof. Otherwise $\text{last}[x] = k$ but then $x = w_k$ by Obs. 3. □

Corollary 7 When line 10 is called, it can be executed.

Proof. This is the exact content of Observation 6. □

Corollary 8 The statement "$x \in U$" is a loop invariant for the **while**-loop in lines 08-12.

Proof. This is immediate from Notation 5 in the light of Obs. 7. □

Observation 9 After line 11 we have $z_\ell = N(z_{\ell-1})$.

Proof. Lines 08-12. □

Observation 10 After line 06,

If $x \in U^*$ then $x \to N(x)$.

Proof. Let $\text{last}[x] = i$. Then, by line 05, $x = w_i$. By the definition of the walk W, there is an edge $w_i \to w_{i+1}$. □

Corollary 11 $[z_0, z_1, \dots]$ is a (finite or infinite) walk.

Proof. Combine Observations 2 and 3. □

Lemma 12 If $x \in U^*$ then $\text{last}[N[x]] > \text{last}[x]$.

Proof. Let $i = \text{last}[x]$. Then $N(x) = w_{i+1}$ and therefore $\text{last}[N(x)] \geq i + 1$. □

Corollary 13 The following is a loop invariant for the **while**-loop 08–11:

$\text{last}[z_0] < \text{last}[z_1] < \cdots < \text{last}[z_\ell]$

Proof. Combine Obs. 9 and Lemma 12. □

Corollary 14 The **while**-loop of lines 08–12 terminates in at most $\min(n, k)$ rounds, and at termination, $z_\ell = w_k$. Moreover, all the z_i are distinct.

Proof. By Cor. 13, the **while**-loop terminates in at most $\min(n, k)$ rounds because for $x \in U^*$ we have $\text{last}[x] \leq k$ and all vertices $\text{last}[z_i]$ are distinct. Moreover, on exit the loop condition must be violated and therefore we have $z_\ell = x = w_k$. Finally, by Corollary 13, this walk has no repetition, because, obviously, if $x = y \in U$ then $\text{last}[x] = \text{last}[y]$. (We also need to note that w_k is not repeated because as soon as we hit $x = w_k$, we exit the **while**-loop of lines 08–12.) □

Theorem 15. $[z_0, \dots, z_\ell]$ is a $u \to \cdots \to v$ path.

Proof. By Corollary 11, $[z_0, z_1, \dots]$ is a finite or infinite walk. By Corollary 14, this walk is finite and ends at $v = w_k$. It begins at $z_0 = w_0$ by line 07. Also by Corollary 14, all the z_i are distinct. Therefore $[z_0, z_1, \dots]$ is a $u \to \cdots \to v$ path. □

COMPLEXITY

We work in the unit cost model. The size of the input is $n + k$. The cost of line 07 and the cost of each execution of our loops is $O(1)$. The **for**-loop of lines 01–03 is executed n times and the **for**-loop of lines 04–06 k times. By Corollary 1, the **while**-loop of lines 08–11 is executed $\ell \leq \min(n, k)$ times; the cost of each execution is $O(1)$. This adds up to $O(n + k)$.