Analysis of Algorithms Homework 4: Savitch's Algorithm

Lowell Batacan, Joshua Steward

November 7, 2016

Savitch's Algorithm is an interesting take on the ever so popular Divide and Conquer algorithm. Divide and Conquer algorithm is an algorith typically seen in Computer Science. Essentially the algorithm recrusivley breaks down a problem into two or more sub-problems until it the problems become simple enough to be solved directly. The solution of the sub-problems are then passed back into the parent problem. Savitch's Algorithm uses the concept of the Divide and Conquer algorithm to solve direct reachibility of a graph starting off with 2 designated points on the graph. The time and space complexity of the algorithm is $O(log^2 n)$.

$$\text{R(G},u, v, i) \Longleftrightarrow (\exists w)\ [\text{R(G},u, w, i\text{-}1) \wedge \text{R(G},w, v, i\text{-}1)]$$

Algorithm Pseudocode

```
1. if  i = 0  then
2.       if  u = v  then
3.             return T
4.        else if  (u,v)  is an edge then
5.             return T
6        end if 7. else
8.        for every vertex w do
9.             if R(G,u, w, i-1) and R(G,w, v, i-1) then
10.                  return T
11.            end if
12.       end for
13. end if
14. return F
```

Our take on Savitchs algorithm is broken up into several parts. First, using a more pythonic approach to populating an n x n matrix, we take input as n and create an n x n matrix from this. The matrix is then initialized, and populated as an adjacency matrix with two built in diagonal paths. The algorithm will search from the top left to the bottom right. As such, Savitchs algorithm

is called with the matrix M as the first input, 0 as the second input, (n * n) 1 or the max distance as the third input, and the size of the matrix as the last input. Savitch begins by appending the first predicate as the current first vertex (u), the current vertex to search for (v) and k to the stack. It then checks for the two base cases of either having the beginning and final vertices being the same or the edge connecting to the last vertex being true, or being 1 as it is in an adjacency matrix. For the recursive call, we search every midpoint w in the path, and call Savitchs algorithm on both halves of the path, with k-1 vertices as the floor for the first recursive call, and k-1 vertices as the ceiling for the second recursive call. This returns true or false, and if true, it proceeds.