Scribe : Sarah PEREZ
Email : sarah.perez@mail.mcgill.ca

Professor Anne CONDON

An introduction to Dynamic Programming : the shortest path algorithm and RNA secondary structure prediction

CPSC 500 - ALGORITHMS AND COMPLEXITY
SCRIBE ASSIGNMENT
Obctober $28^{th}2013$

Overview

Dynamic programming solves complex problems by dividing them into sub-problems. Unlike "divide and conquer" algorithms, the sub-problems often overlap in their solution's search space and the number of sub-problems is typically polynomial. Two important algorithms are presented here to illustrate the efficiency of dynamic programming methods. First the shortest path problem is solved using the Floyd-Warshall algorithm. Second, a basic RNA secondary structure prediction algorithm is presented. Finally, dynamic programming based algorithms allow for the search of sub-optimal solutions, a topic which is briefly introduced.

1 Shortest Path Algorithm

Given a directed graph with weighted edges between the nodes of the path, we wish to find the shortest path between two arbitrary nodes. This algorithm has been adapted to real life problems such as the development of Google Maps [1].

> **Note :** We assume that the graph doesn't contain any negative cycles which, when travelled, would reduce the path length and could in principle be travelled an infinite number of times.

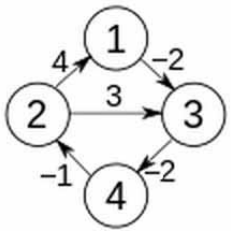

FIGURE 1 – An example of a directed graph with a negative cycle.

As with most problems which can be solved using dynamic programming, we first make a few observations related to the problem statement. We then identify a quantity which once calculated, minimized or maximized, provides an optimal solution. A recursion is then developed to formalize this quantity mathematically. The algorithm specifies the procedure used to calculate the quantity for sub-problems of the original problem.

The Floyd-Warshall Algorithm

Our first observation is that the shortest path between two nodes i and j could simply be (i, j) if they are neighbouring nodes, otherwise it goes through intermediate nodes. We shall denote as F_{ij} the weighted length of the shortest path between two nodes i and j. Let k be the largest index of a node on such a shortest path, then $F_{ij} = F_{ik} + F_{kj}$ and the shortest path from $i \to k$ and $k \to j$ only visits nodes between 1 and $k - 1$.

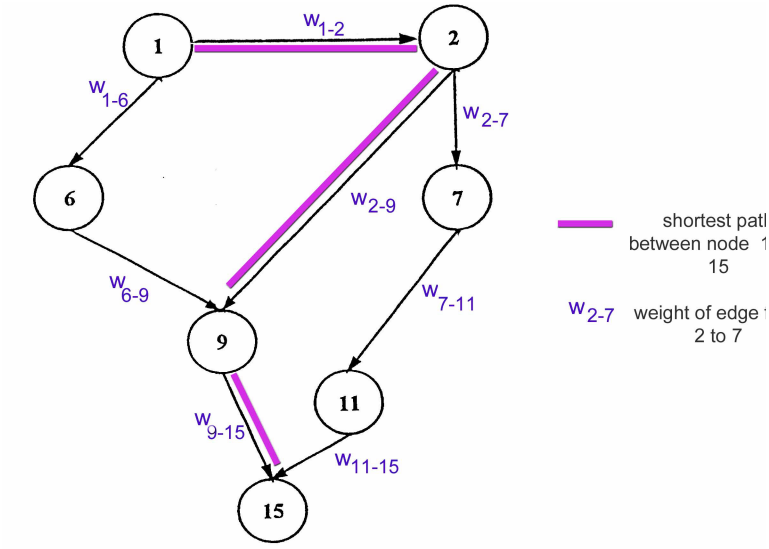

FIGURE 2 – An example of a directed graph with different paths going from node 1 to 15.

Note : The node k can't appear twice on a shortest path since that would involve having a cycle since there are no negative cycles and there is really no advantage in travelling on a positive cycle.

Let F_{ij}^k be the length of shortest path from i to j that only visits intermediate nodes with index $\leq k$. We now consider the base and general cases for the recursion for a graph of size n.

BASE CASE :

$$F_{ij}^0 = \begin{cases} w_{ij}, & \text{if edge (i,j) is in the graph.} \\ \infty, & \text{otherwise.} \end{cases} \tag{1}$$

GENERAL CASE :
$1 \leq i, j, k \leq n$

$$F_{ij}^k = min \begin{cases} F_{ik}^{k-1} + F_{kj}^{k-1}, & \text{if shortest path goes through } k. \\ F_{ij}^{k-1}, & \text{otherwise.} \end{cases} \tag{2}$$

The problem can be visualized as a 3D matrix containing the values of the F_{ij}^k for all possible combinations of i, j, k. The running time of this algorithm is then at most $O(n^3)$.

2 RNA secondary structure prediction

Ribonucleic acid (RNA) are biological molecules which play a major role in the coding, regulation and expression of genes in a cell. They are assembled in a chain of nucleotides which can be of four different types : Adenosine, Cytosine, Uracil, and Guanine. The nucleotides bind in pairs to give the molecule a secondary structure which we wish to predict.

Motivation

RNA's secondary structure minimizes the molecule's free energy and stabilizes it. Predicting an RNA molecule's secondary structure is particularly important in the study of protein-RNA or DNA-RNA binding for the secondary structure creates functional binding domains for the molecule (Figure 3).

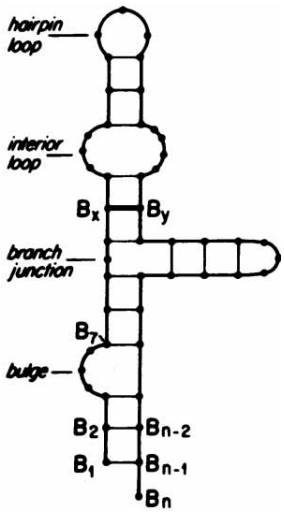

FIGURE 3 – An example of an RNA secondary structure with labelled bases B_1 to B_n and different structural domains[2].

Problem formulation

Given a sequence $S = s_1, s_2, \ldots, s_n$ of RNA over the alphabet $\Sigma = A, U, C, G$, a secondary structure is a set of pairs i-j with $1 \leq i, j \leq n$. We wish to find the secondary structure with the maximum number of base pairs. The following pairs are allowed :

1. no index appears in more than 1 pair,
2. pairs don't cross and create **pseudoknots** : No pairs i-j and i'-j' with $i < i' < j < j'$,
3. For all pairs i-j then i < j +3,
4. For all pairs i-j they are in the set s_i-$s_j \in \{A\text{-}U, C\text{-}G\}$.

The algorithm

The quantity we wish to maximize is the number of pairs between a substring s_i to s_j which we will denote as $F(i, j)$. The smallest substring with one pair is a substring of length 4 because of condition 3.

BASE CASE :

$i + 4 = j$

$$F(i,j) = \begin{cases} 1, & \text{if } (s_i, s_j) \text{ match according to condition 4} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

For larger substrings, if the pair (s_i, s_j) is valid then we match s_i with s_j and continue the procedure with the smaller substring s_{i+1} to s_{j-1}. Otherwise, we find the break point k to split the substring into two smaller substrings and continue the recursion.

GENERAL CASE :

$i + 4 < j$

$$F(i,j) = max \begin{cases} F(i+1, j-1) + 1, & \text{if } (s_i, s_j) matchaccordingtocondition4 \\ \max\limits_{i \leq k < j} F(k+1, j) + F(i, k), & \text{otherwise.} \end{cases} \tag{4}$$

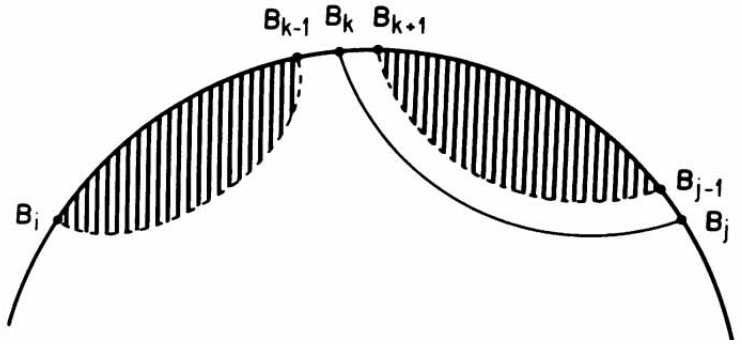

FIGURE 4 – An illustration of the different possible cases for $F(i, j)$[2].

Once all the $F(i, j)$ are calculated we can determine a plausible secondary structure as well as the number of pairs in our sequence from $F(1, n)$. The running time of this algorithm is $O(n^3)$.

3 Finding sub-optimal solutions

An interesting advantage of using dynamic programming based algorithms is that a sub-optimal solution can easily be found. These solutions are particularly useful when modelling biological process such as molecular folding which are difficult to predict and where multiple solutions are often considered and further tested. For example, Waterman and Byers modified the shortest path algorithm to find neighbouring solutions to the optimum by a specified distance [3]. They motivate the use of their algorithm by demonstrating its application to analyzing evolutionary distant DNA sequences. Certain dynamic programming algorithms don't need to be modified to find sub-optimal solutions : at dynamic table created from the recursion can be traversed to pick out neighbouring solutions. For example the 3D table created in either of the problems defined above can be read such that certain pairings or certain paths are not taken.

4 Conclusion

Two classic examples of dynamic programing algorithms were presented : the shortest path problem in a directed graph and the RNA secondary structure prediction problem. In each case we quantified a key variable in obtaining the solution and wrote a recursive procedure to calculate its value in sub-problems. Base cases and general cases were defined to obtain the final solution. In addition, different sub-optimal finding methods were introduced.

References

[1] Sanders, Peter (March 23, 2009). *Fast route planning*. Google Tech Talk.

[2] Nussinov, Ruth and Jacobson, Ann B. (1980) *Fast algorithm for predicting the secondary structure of single-stranded RNA*. Proc. nat. Acad. Sci. USA 77, 6309-6313.

[3] Waterman, Michael S. and Byers, Thomas H. (1985) *A Dynamic Programming Algorithm to Find All Solutions in a Neighborhood of the Optimum*. Mathematical Biosciences 77,179-188.