Algorithm Design XI

Dynamic Programming II

Guoqiang Li
School of Software

SHANGHAI JIAO TONG UNIVERSITY

Past Exam

2.　（15 分）考虑**三分问题（3-Partition problem）**：给定整数集合$\{a_1, a_2, \dots, a_n\}$，判断是否可以将其分割为三个相互不相交的子集I, J, K，使得

$$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \sum_{k \in K} a_i = \frac{1}{3} \sum_{i=1}^{n} a_i$$

6． (20 分)在一条河上有一座独木桥，长度为 L，上面分布着一些石子，为了简单起见，我们假设桥为 $0\text{-}L$ 的一段线段，而石子都分布在整数坐标上，也就是有一个函数 *stone(x)*，表示在坐标 x 上是否有石子，比如 *stone(0)= 1* 表示在桥头有一个石子，*stone(2) = 0* 表示在坐标为 *2* 的位置没有石子。

现在有一个小朋友站在桥头的位置想要过桥（站在桥尾或者跨过桥尾均为过了桥），但他不想踩到石子，他每跨出一步的步长是 $[S,T]$ 区间中的任何整数（包括 S 和 T）。设计算法求小朋友要过河，必须踩到的最少的石子数。

（1）（15 分）写出动态规划范式，注意边界条件，并详细说明动态规划范式所代表的意思。

（2）（5 分）根据动态规划范式，给出时间空间复杂度分析。

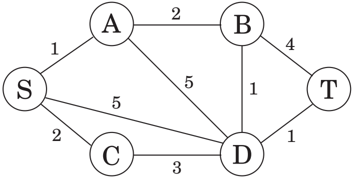
Shortest Reliable Paths

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra "hop" fraught with uncertainties and dangers of packet loss.

We would like to avoid paths with too many edges.



Shortest Reliable Paths

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k, and we want the shortest path from s to t that uses at most k edges.

Dynamic programming will work!

Dynamic Programming

For each vertex v and each integer $i \leq k$, let

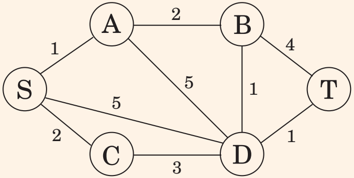
$$dist(v,i) = \text{ the length of the shortest path from } s \text{ to } v \text{ that uses } i \text{ edges}$$

The starting values $dist(v,0)$ are ∞ for all vertices except s, for which it is 0.

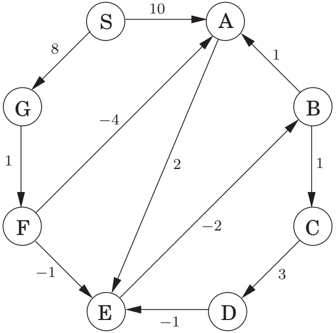
$$dist(v,i) = \min_{(u,v)\in E}\{dist(u,i-1) + l(u,v)\}$$

Shortest Reliable Paths

Find out the shortest reliable path from S to T, when $k = 3$.



Bellman-Ford Algorithm



	Iteration							
Node	0	1	2	3	4	5	6	7
S	0	0	0	0	0	0	0	0
A	∞	10	10	5	5	5	5	5
B	∞	∞	∞	10	6	5	5	5
C	∞	∞	∞	∞	11	7	6	6
D	∞	∞	∞	∞	∞	14	10	9
E	∞	∞	12	8	7	7	7	7
F	∞	∞	9	9	9	9	9	9
G	∞	8	8	8	8	8	8	8

All-Pairs Shortest Path

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

One approach would be to execute Bellman-Ford-Moore algorithm $|V|$ times, once for each starting node.

The total running time would then be $O(|V|^2|E|)$.

We'll now see a better alternative, the $O(|V|^3)$, named Floyd-Warshall algorithm.

Floyd-Warshall Algorithm

Dynamic programming again!

The Subproblems

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k) =$ the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

Initially, $dist(i, j, 0)$ is the length of the direct edge between i and j, if it exists, and is ∞ otherwise.

For $k \geq 1$

$$dist(i, j, k) = \min\{dist(i, j, k-1), dist(i, k, k-1) + dist(k, j, k-1)\}$$

The Program


```
for i = 1 to n do
    for j = 1 to n do
        dist(i, j, 0) = ∞;
    end
end
for all (i, j) ∈ E do
    dist(i, j, 0) = l(i, j);
end
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            dist(i, j, k) = min{dist(i, j, k − 1), dist(i, k, k − 1) + dist(k, j, k − 1)};
        end
    end
end
```

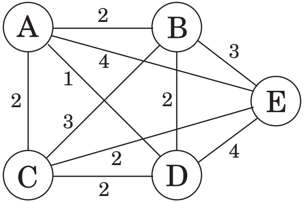
Traveling Salesman Problem

The Traveling Salesman Problem

A traveling salesman is getting ready for a big sales tour. Starting at his hometown, he will conduct a journey in which each of his target cities is visited exactly once before he returns home.

Q: Given the pairwise distances between cities, what is the best order in which to visit them, so as to minimize the overall distance traveled?

The brute-force approach is to evaluate every possible tour and return the best one. Since there are $(n-1)!$ possibilities, this strategy takes $O(n!)$ time.



The Traveling Salesman Problem

Denote the cities by $1, \ldots, n$, the salesman's hometown being 1, and let $D = (d_{ij})$ be the matrix of intercity distances.

The goal is to design a tour that starts and ends at 1, includes all other cities exactly once, and has minimum total length.

The Subproblems

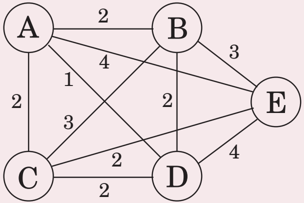
For a subset of cities $S \subseteq \{1, 2, \ldots, n\}$ that includes 1, and $j \in S$, let $C(S, j)$ be the length of the shortest path visiting each node in S exactly once, starting at 1 and ending at j.

When $|S| > 1$, we define $C(S, 1) = \infty$.

For $j \neq 1$ with $j \in S$ we have

$$C(S, j) = \min_{i \in S : i \neq j} C(S \backslash \{j\}, i) + d_{ij}$$

Exercise



The Program

$C(1,1) = 0$;
for $s = 2$ *to* n **do**
  for *all subsets* $S \subseteq \{1, 2, \ldots, n\}$ **do**
    $C(S,1) = \infty$;
    for *all* $j \in S$ *and* $j \neq 1$ **do**
      $C(S,j) = \min_{i \in S: i \neq j} C(S \backslash \{j\}, i) + d_{ij}$;
    end
  end
end
`return`($\min_j C(\{1, 2, \ldots, n\}, j) + d_{j1}$);

There are at most $2^n \cdot n$ subproblems, and each one takes linear time.

The total running time is therefore $O(n^2 \cdot 2^n)$.

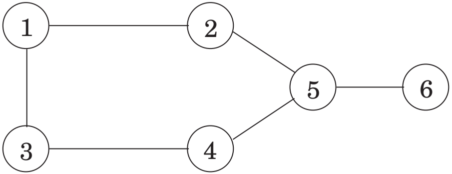
Independent Sets in Trees

The Problem

A subset of nodes $S \subseteq V$ is an independent set of graph $G = (V, E)$ if there are no edges between them.

Finding the largest independent set in a graph is believed to be intractable.

However, when the graph happens to be a tree, the problem can be solved in linear time, using dynamic programming.



The Subproblems

Shanghai Jiao Tong University

$I(u)$ = size of largest independent set of subtree hanging from u.

$$I(u) = \max\{1 + \sum_{\text{grandchildren } w \text{ of } u} I(w), \sum_{\text{children } w \text{ of } u} I(w)\}$$