codeforces.com

A little bit of classics: dynamic programming over subsets and paths in graphs

7-9 minutos

Author thanks adamax *for translation this article into English.*

Introduction

After Codeforces Beta Round #11 several participants expressed a wish to read something about problems similar to problem D of that round. The author of this article, for the purpose of helping them, tried searching for such information, but to his surprise couldn't find anything on the Internet. It is not known whether the search was not thorough or there's really nothing out there, but (just in case) the author decided to write his own article on this topic.
In some sense this article may be regarded as a tutorial for the problem D from Beta Round #11.

In this article quite well-known algorithms are considered: the search for optimal Hamiltonian walks and cycles, finding their number, check for existence and something more. The so-called "dynamic programming over subsets" is used.

This method requires exponential time and memory and therefore can be used only if the graph is very small - typically 20 vertices or less.

DP over subsets

Consider a set of elements numbered from 0 to $N - 1$. Each subset of this set can be encoded by a sequence of N bits (we will call this sequence "a mask"). The i-th element belongs to the subset if and only if the i-th bit of the mask equals 1. For instance, the mask 00010011 means that the subset of the set [0... 7] consists of elements 0, 1 and 4. There are totally 2^N masks, and so 2^N subsets. Each mask is in fact an integer number written in binary notation.
The method is to assign a value to each mask (and, therefore, to each subset) and compute the values for new masks using already computed values. As a rule, to find the value for a subset A we remove an element in every possible way and use values for obtained subsets A'_1, A'_2, ... , A'_k to compute the value for A. This means that the values for A_i' must have been computed already, so we need to establish an ordering in which masks will be considered. It's easy to see that the natural ordering will do: go over masks in increasing order of corresponding numbers.

We will use the following notation:

bit(*i*, *mask*) - the *i*-th bit of *mask*
count(*mask*) - the number of non-zero bits in *mask*
first(*mask*) - the number of the lowest non-zero bit in *mask*

(*a*?*b*: *c*) - returns *b* if *a* holds, or *c* otherwise.The elements of our set will be vertices of the graph. For the sake of simplicity we'll assume that the graph is undirected. Modification of the algorithms for directed graphs is left as an exercise for the reader.

1. Search for the shortest Hamiltonian walk

Let the graph $G = (V, E)$ have n vertices, and each edge $(i, j) \in E$ have a weight $d(i, j)$. We want to find a Hamiltonian walk for which the sum of weights of its edges is minimal. Let $dp[mask][i]$ be the length of the shortest Hamiltonian walk in the subgraph generated by vertices in *mask*, that ends in the vertex *i*.

The DP can be calculated by the following formulas:

$dp[mask][i] = 0$, if $count(mask) = 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = \min_{bit(j,mask)=1,(j,i)\in E}\{dp[mask \text{ xor } 2^i][j] + d(j,i)\}$, if $count(mask) > 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = \infty$ in other cases.
Now the desired minimal length is $p_{min} = \min_{i\in[0\cdots n-1]}\{dp[2^n - 1][i]\}$. If $p_{min} = \infty$, then there is no Hamiltonian walk in the graph. Otherwise it's easy to recover the walk itself. Let the minimal walk end in the vertex *i*. Then the vertex $j \neq i$, for which $dp[2^n - 1][i] = dp[2^n - 1 \text{ xor } 2^i][j] + d(j,i)$, is the previous vertex in the path. Now remove *i* from the set and find the vertex previous to *j* in the same way. Continuing this process until only one vertex is left, we'll find the whole Hamiltonian walk.

This solution requires $O(2^n n)$ of memory and $O(2^n n^2)$ of time.

2. Finding the number of Hamiltonian walks

Let the graph $G = (V, E)$ be unweighted. We'll modify the previous algorithm. Let $dp[mask][i]$ be the number of Hamiltonian walks on the subset $mask$, which end in the vertex i. The DP is rewritten in the following way:
$dp[mask][i] = 1$, if $count(mask) = 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = \sum_{bit(j,mask)=1,(j,i)\in E} dp[mask \text{ xor } 2^i][j]$, if $count(mask) > 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = 0$ in other cases.
The answer is
$\sum_{i\in[0\cdots n-1]} dp[2^n - 1][i]$
.

This solution requires $O(2^n n)$ of memory and $O(2^n n^2)$ of time.

3. Finding the number of simple paths

Calculate the DP from the previous paragraph. The answer is $1/2 \sum_{i\in[0\cdots n-1],count(mask)\geqslant 2} dp[mask][i]$. The coefficient $1 / 2$ is required because each simple path is considered twice - in both directions. Also note that only paths of positive length are taken into account. You can add n zero-length paths, of course.
This solution requires $O(2^n n)$ of memory and $O(2^n n^2)$ of time.

4. Check for existence of Hamiltonian walk

We can use solution 2 replacing the sum with bitwise OR.

$dp[mask][i]$ will contain a boolean value - whether there exists a Hamiltonian walk over the subset *mask* which ends in the vertex *i*. DP is the following:
$dp[mask][i] = 1$, if $count(mask) = 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = \mathrm{OR}_{bit(j,mask)=1,(j,i)\in E} dp[mask \text{ xor } 2^i][j]$, if $count(mask) > 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = 0$ in other cases.
This solution, like solution 2, requires $O(2^n n)$ of memory and $O(2^n n^2)$ of time. It can be improved in the following way.

Let $dp'[mask]$ be the mask of the subset consisting of those vertices *j* for which there exists a Hamiltonian walk over the subset *mask* ending in *j*. In other words, we 'compress' the previous DP: $dp'[mask]$ equals $\sum_{i\in[0\cdots n-1]} dp[mask][i] \cdot 2^i$. For the graph *G* write out *n* masks M_i, which give the subset of vertices incident to the vertex *i*. That is, $M_i = \sum_{j\in[0\cdots n-1]} 2^j \cdot ((i,j) \in E?1:0)$.

DP will be rewritten in the following way:

$dp'[mask] = 2^i$, if $count(mask) = 1$ and $bit(i, mask) = 1$;
$dp'[mask] = \sum_{i\in[0\cdots n-1]} 2^i \cdot ((dp[mask \text{ xor } 2^i] \text{ and } M_i) \neq 0?1:0)$, if $count(mask) > 1$;
$dp'[mask] = 0$ in other cases.
Pay special attention to the expression $dp[mask \text{ xor } 2^i] \text{ and } M_i$. The first part of the expression is the subset of vertices *j*, for which there exists a Hamiltonian walk over the subset *mask* minus vertex *i*, ending in *j*. The second part of the expression is the set of vertices incident

to *i*. If these subsets have non-empty intersection (their bitwise AND is non-zero), then it's easy to see that there exists a Hamiltonian walk in *mask* ending in the vertex *i*.

The final test is to compare $dp[2^n - 1]$ to 0.

This solution uses $O(2^n)$ of memory and $O(2^n n)$ of time.

5. Finding the shortest Hamiltonian cycle

Since we don't care at which vertex the cycle starts, assume that it starts at 0. Now use solution 1 for the subset of vertices, changing the formulas in the following way:
$dp[1][0] = 0$;
$dp[mask][i] = \min_{bit(j,mask)=1,(j,i)\in E}\{dp[mask \text{ xor } 2^i][j] + d(j,i)\}$, if $i > 0$, $bit(0, mask) = 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = \infty$ in other cases.
So $dp[mask][i]$ contains the length of the shortest Hamiltonian walk over the subset *mask*, starting at 0 and ending at *i*.

The required minimum is calculated by the formula $min_{i\in[1\cdots n-1],(i,0)\in E}\{dp[2^n-1][i] + d(i,0)\}$. If it equals ∞, there is no Hamiltonian cycle. Otherwise the Hamiltonian cycle can be restored by a method similar to solution 1.

6. Finding the number of Hamiltonian cycles

Using ideas from solutions 5 and 2 one can derive a DP calculating the number of Hamiltonian cycles requiring $O(2^n n^2)$ of time and $O(2^n n)$ of memory.

7. Finding the number of simple cycles

Let $dp[mask][i]$ be the number of Hamiltonian walks over

the subset *mask*, starting at the vertex *first*(*mask*) and ending at the vertex *i*. DP looks like this:
$dp[mask][i] = 1$, if $count(mask) = 1$ and $bit(i, mask) = 1$;
$dp[mask][i] = \sum_{j \in [0 \cdots n-1], (j,i) \in E} dp[mask \text{ xor } 2^i][j]$, if $count(mask) > 1$, $bit(i, mask) = 1$ and $i \neq first(mask)$;
$dp[mask][i] = 0$ otherwise.
The answer is
$1/2 \sum_{i \in [0 \cdots n-1], count(mask) \geqslant 3, (first(mask), i) \in E} dp[mask][i]$
.

This solution requres $O(2^n n^2)$ of time and $O(2^n n)$ of memory.

8. Checking for existence of Hamiltonian cycle

We can modify solution 5 and, using the trick from solution 4, obtain an algorithm requiring $O(2^n n)$ of time and $O(2^n)$ of memory.
Exercises

CFBR11D
CCTOOLS
P.S. This article may be extended and fixed in future. The author would be grateful for supplementing the Exercises section and for pointing out any mistakes and inaccuracies.