數位三 s10755014 鄭世傑

問題：Steiner Tree[[1]](#footnote-1)[1]（史坦納樹）

問題敘述：

在一個給定的無向圖中，頂點之間有路相連並帶有權重，所有的頂點的集合為S，如果給定一個集合K為S的子集，能夠找到一個樹將K所有的點串聯起來（也可以有K以外的點）並使得權重為最少，則稱此樹為Steiner Tree。

問題分類[[2]](#footnote-2)[2]：

根據程式碼所算出來的時間複雜度為O(n 3^t + n^2 2^t + n^3)，可得知此問題為NP-hard問題，因為它無法在多項式時間內找到解。但是如果是尋找權重最低的樹的話為NP-complete問題[[3]](#footnote-3)。

應用[[4]](#footnote-4)[4]：

史坦納樹由數學延伸而來，之後被廣泛應用在計算機科學領域，如電路板、積體電路等路徑問題，一般在電子領域當中規劃路徑，並把多組路徑合併來改良連結路徑。

問題解法：[2]

使用Floyd Warshall計算所有路徑d並將結果存入矩陣，管理Steiner Tree集合S為OPT[S][p]，其中P為端點，之後執行以下兩個命令：

1. | S | = {p} OPT [S] [q] = d [p] [q]
2. | S |當≧2 OPT [S] [p] = min\_ {q \ in V，E \ subset S}（d [p，q] + OPT [E] [q] + OPT [SE] [q] ）

第二項命令將S分為兩個部分，以q為末端計算，並附加p—q以p為末端計算，獲得結果後將T分為兩個部分，找到所有以q為末端的樹，以q為跟連接即可得到T的Steiner Tree。

程式碼：

weight minimum\_steiner\_tree(const vector<int>& T, const matrix &g) {

const int n = g.size();

const int numT = T.size();

if (numT <= 1) return 0;

matrix d(g); // all-pair shortest

for (int k = 0; k < n; ++k)

for (int i = 0; i < n; ++i)

for (int j = 0; j < n; ++j)

d[i][j] = min( d[i][j], d[i][k] + d[k][j] );

weight OPT[(1 << numT)][n];

for (int S = 0; S < (1 << numT); ++S)

for (int x = 0; x < n; ++x)

OPT[S][x] = inf;

for (int p = 0; p < numT; ++p) // trivial case

for (int q = 0; q < n; ++q)

OPT[1 << p][q] = d[T[p]][q];

for (int S = 1; S < (1 << numT); ++S) { // DP step

if (!(S & (S-1))) continue;

for (int p = 0; p < n; ++p)

for (int E = 0; E < S; ++E)

if ((E | S) == S)

OPT[S][p] = min( OPT[S][p], OPT[E][p] + OPT[S-E][p] );

for (int p = 0; p < n; ++p)

for (int q = 0; q < n; ++q)

OPT[S][p] = min( OPT[S][p], OPT[S][q] + d[p][q] );

}

weight ans = inf;

for (int S = 0; S < (1 << numT); ++S)

for (int q = 0; q < n; ++q)

ans = min(ans, OPT[S][q] + OPT[((1 << numT)-1)-S][q]);

}

/\* 以下は PKU 3123 用

weight ans = inf;

for (int S = 0; S < (1 << numT); ++S) {

weight sub = 0;

for (int P = 0; P < 4; ++P) {

int E = 0, p = -1;

for (int k = 0; k < 8; k += 2)

if (((S >> k) & 3) == P)

E += 3 << k, p = T[k];

sub += (!!E) \* OPT[E][p];

}

ans = min(ans, sub);

}

return ans;

\*/weight minimum\_steiner\_tree(**const** vector<**int**>& T, **const** matrix &g) {

**const** **int** n = g.size();weight minimum\_steiner\_tree(const vector<int>& T, const matrix &g) {

const int n = g.size();

const int numT = T.size();

if (numT <= 1) return 0;

matrix d(g); // all-pair shortest

for (int k = 0; k < n; ++k)

for (int i = 0; i < n; ++i)

for (int j = 0; j < n; ++j)

d[i][j] = min( d[i][j], d[i][k] + d[k][j] );

weight OPT[(1 << numT)][n];

for (int S = 0; S < (1 << numT); ++S)

for (int x = 0; x < n; ++x)

OPT[S][x] = inf;

for (int p = 0; p < numT; ++p) // trivial case

for (int q = 0; q < n; ++q)

OPT[1 << p][q] = d[T[p]][q];

for (int S = 1; S < (1 << numT); ++S) { // DP step

if (!(S & (S-1))) continue;

for (int p = 0; p < n; ++p)

for (int E = 0; E < S; ++E)

if ((E | S) == S)

OPT[S][p] = min( OPT[S][p], OPT[E][p] + OPT[S-E][p] );

for (int p = 0; p < n; ++p)

for (int q = 0; q < n; ++q)

OPT[S][p] = min( OPT[S][p], OPT[S][q] + d[p][q] );

}

weight ans = inf;

for (int S = 0; S < (1 << numT); ++S)

for (int q = 0; q < n; ++q)

ans = min(ans, OPT[S][q] + OPT[((1 << numT)-1)-S][q]);

}

/\* 以下は PKU 3123 用

weight ans = inf;

for (int S = 0; S < (1 << numT); ++S) {

weight sub = 0;

for (int P = 0; P < 4; ++P) {

int E = 0, p = -1;

for (int k = 0; k < 8; k += 2)

if (((S >> k) & 3) == P)

E += 3 << k, p = T[k];

sub += (!!E) \* OPT[E][p];

}

ans = min(ans, sub);

}

return ans;

\*/

**const** **int** numT = T.size();

**if** (numT <= 1) **return** 0;

matrix d(g); // all-pair shortest

**for** (**int** k = 0; k < n; ++k)

**for** (**int** i = 0; i < n; ++i)

**for** (**int** j = 0; j < n; ++j)

d[i][j] = min( d[i][j], d[i][k] + d[k][j] );

weight OPT[(1 << numT)][n];

**for** (**int** S = 0; S < (1 << numT); ++S)

**for** (**int** x = 0; x < n; ++x)

OPT[S][x] = inf;

**for** (**int** p = 0; p < numT; ++p) // trivial case

**for** (**int** q = 0; q < n; ++q)

OPT[1 << p][q] = d[T[p]][q];

**for** (**int** S = 1; S < (1 << numT); ++S) { // DP step

**if** (!(S & (S-1))) **continue**;

**for** (**int** p = 0; p < n; ++p)

**for** (**int** E = 0; E < S; ++E)

**if** ((E | S) == S)

OPT[S][p] = min( OPT[S][p], OPT[E][p] + OPT[S-E][p] );

**for** (**int** p = 0; p < n; ++p)

**for** (**int** q = 0; q < n; ++q)

OPT[S][p] = min( OPT[S][p], OPT[S][q] + d[p][q] );

}

weight ans = inf;

**for** (**int** S = 0; S < (1 << numT); ++S)

**for** (**int** q = 0; q < n; ++q)

ans = min(ans, OPT[S][q] + OPT[((1 << numT)-1)-S][q]);

}

/\* 以下は PKU 3123 用

weight ans = inf;

for (int S = 0; S < (1 << numT); ++S) {

weight sub = 0;

for (int P = 0; P < 4; ++P) {

int E = 0, p = -1;

for (int k = 0; k < 8; k += 2)

if (((S >> k) & 3) == P)

E += 3 << k, p = T[k];

sub += (!!E) \* OPT[E][p];

}

ans = min(ans, sub);

}

return ans;

\*/

1. <https://en.wikipedia.org/wiki/Steiner_tree_problem> [↑](#footnote-ref-1)
2. <http://www.prefield.com/algorithm/dp/steiner_tree.html> [↑](#footnote-ref-2)
3. <http://dspace.lib.fcu.edu.tw/bitstream/2377/30080/1/CEV%202-1.pdf> [↑](#footnote-ref-3)
4. <http://web.ntnu.edu.tw/~algo/SpanningTree2.html#4> [↑](#footnote-ref-4)