MinCostMatching

```
// Min cost bipartite matching via
shortest augmenting paths. This is
an O(n^3) implementation of a
shortest augmenting path algorithm
for finding min cost perfect
matchings in dense graphs.
//   cost[i][j] = cost for pairing
left node i with right node j
//   Lmate[i] = index of right node
that left node i pairs with
//   Rmate[j] = index of left node
that right node j pairs with
// The values in cost[i][j] may be
positive or negative.  To perform
// maximization, simply negate the
cost[][] matrix.
#include <algorithm>
#include <cstdio>
#include <cmath>
#include <vector>
using namespace std;
typedef vector<double> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;
double MinCostMatching(const VVD
&cost, VI &Lmate, VI &Rmate) {
  int n = int(cost.size());
  VD u(n);VD v(n);
  for (int i = 0; i < n; i++) {
    u[i] = cost[i][0];
    for (int j = 1; j < n; j++)
u[i] = min(u[i], cost[i][j]);
  }
  for (int j = 0; j < n; j++) {
    v[j] = cost[0][j] - u[0];
    for (int i = 1; i < n; i++)
v[j] = min(v[j],cost[i][j]- u[i]);}
  Lmate = VI(n, -1);Rmate =VI(n,-
1);int mated = 0;
  for (int i = 0; i < n; i++) {
  for (int j = 0; j < n; j++) {if
(Rmate[j]!=-1)continue;
if (fabs(cost[i][j]-u[i]-v[j])<1e-
10) {Lmate[i]=j;Rmate[j]=i;mated++;
break;}}}
VD dist(n);VI dad(n);VI seen(n);
while(mated<n){int s=0;
   while (Lmate[s] != -1) s++;
  fill(dad.begin(), dad.end(), -1);
 fill(seen.begin(), seen.end(), 0);
    for (int k = 0; k < n; k++)
      dist[k] = cost[s][k] - u[s] -
v[k];

    int j = 0;
    while (true) {

      // find closest
      j = -1;
      for (int k = 0; k < n; k++) {
        if (seen[k]) continue;
        if (j == -1 || dist[k] <
dist[j]) j = k;
      }
      seen[j] = 1;

      // termination condition
      if (Rmate[j] == -1) break;


      const int i = Rmate[j];
      for (int k = 0; k < n; k++) {
        if (seen[k]) continue;
        const double new_dist =
dist[j] + cost[i][k] - u[i] - v[k];
        if (dist[k] > new_dist) {
          dist[k] = new_dist;
          dad[k] = j;}}}

    for (int k = 0; k < n; k++) {
      if (k == j || !seen[k])
continue;
      const int i = Rmate[k];
      v[k] += dist[k] - dist[j];
      u[i] -= dist[k] - dist[j];
    }
    u[s] += dist[j];

    while (dad[j] >= 0) {
      const int d = dad[j];
      Rmate[j] = Rmate[d];
      Lmate[Rmate[j]] = j;
      j = d;}
  Rmate[j]=s;Lmate[s] = j;mated++;}
  double value = 0;
  for (int i = 0; i < n; i++)
    value += cost[i][Lmate[i]];

  return value;}
```