C++

// zbiór plut

std::unordered_map<std::vector<string>,std::unordered_map<std::vector<string>,double>> d;

macierz

	A	B	C
A			
B			d_{ij}
C			

for i ← 1 to n do
  for j ← 1 to n do
    d[i][j] ← metryka(i,j); // d->first = int(i); d->second->first =
  end for
end for

while (d.size() > 1)

- wyszukaj minimum
- nowy wiersz a ← (a,b), to samo kolumny

for i ← 1 to n do

d[i][a,b] ← d[a,b][i] ← min(d[a][i], d[b][i]) // single

d[i][a,b] ← d[a,b][i] ← max(d[a][i], d[b][i]) // complete link

---

// std::unordered_map<std::vector<std::string>> min

min[c]

```
for (int i=0; i<d.size(); i++)
    for (int j=0; j<d.size(); j++) {
        d[i]->second[j]->second = metryka[i][j];
        if( d[i]->second[min[i]]->second > d[i]->second[j]->second )
            min[i]->first = d[i]->second->first;
    }
```

// wyszukiwanie odległości z minimum z i do x

// main_loop -> klastrowanie

```
for(int s=0; s<d.size(); s++)
        <std::vector<string>> s1;
```

dynamic

// najbliższe pary

```
for (int i=0; i<d.size(); i++)
    if (d[i]->second[min[i]]->second < d[s1]->second[s1]->second)
        s1 = d[i]->second[i]->first;
```

// indeks 1. elementu z pary najbliższej

```
std::vector<std::string> s2 = min[s1]
auto data1 = d.search(s1); auto data2 = d.search(s2);
// redukcja kolumn
for (int j=0; j<d.size(); j++)
    if (data2[j]->second < data1[j]->second)
        data3 = d.search(j);
    data1[j]->second = (data3.search(data1))->second = data2[j]->second;
data1->first.append(s2); // dodać porost zmiany (konkatenacje)
+ wyrzucić element data 2
// tablica min
for (int i=0; i<d.size(); i++)
    if(min[i] == data2)
        min[i] = data1;
    if(data1[i]->second < data1[min.search(data1)]->second)
        min[data1] = d[i]->first;
```