1. Chamfer Distance between Sphere Sets

Chamfer Distance는 간단히 (sphere_tree A의 각각의 sphere center들로부터 가장 가까운 sphere_tree B의 sphere center까지의 distance 합) + (sphere_tree B의 각각의 sphere center들로부터 가장 가까운 sphere_tree A의 sphere center까지의 distance 합)으로 구할 수 있습니다.

$$A = \{x_1, x_2, \ldots\} \, (x_i : center\ of\ sphere)$$
$$B = \{y_1, y_2, \ldots\} \, (y_i : center\ of\ sphere)$$
$$CD = \sum_i \|x_i - y_{\min}\| + \sum_i \|y_i - x_{\min}\|$$

($y_{\min}$:closest point in B from x_i, $x_{\min}$: closest point in A from y_i)

```
CD (sphere_tree A, sphere_tree B, level k){
  cd=0;
  for each sphere x in A in level k{
    cd+=find_minimum_distance(x,B);
  }
  for each sphere y in B in level k{
    cd+=find_minimum_distance(y,A);
  }
  return cd;
}

find_minimum_distance(sphere x, sphere_tree A){
  min_d=0;
  for each sphere y in A{
    min_d = min(min_d, distance(x,y));
  }
  return min_d
}
```

2. EMD between Sphere Sets

EMD는 두 확률 분포 사이의 차이를 계산할 때 유용한데, 그래프로 옮겨보면 다음처럼 단순화시킬 수 있습니다.

set A = $\{v_1, v_2, \dots\}$, set B = $\{u_1, u_2, \dots\}$ (A, B : set of vertices)

A의 각 vertex 들은 $w(v_i)$의 값을 가지고 있고, (양수라고 가정)

B의 각 vertex 들은 $w(u_i)$의 값을 가지고 있다.

전제조건 : $\sum_i w(v_i) = \sum_j w(u_j)$ (확률 분포의 경우는 1로 동일한 값을 가질 것입니다.)

(graph의 network-flow 문제와 매우 유사합니다.)

v_i와 u_j를 잇는 edge는 weight d_{ij}를 갖는다. (우리의 경우 distance로 정의)

work function을 다음과 같이 정의하는데,

$$W(A, B, F) = \sum_i \sum_j f_{ij} d_{ij}$$

(단, $F = [f_{ij}]$는 v_i와 u_j사이의 edge에서의 flow를 의미한다.)

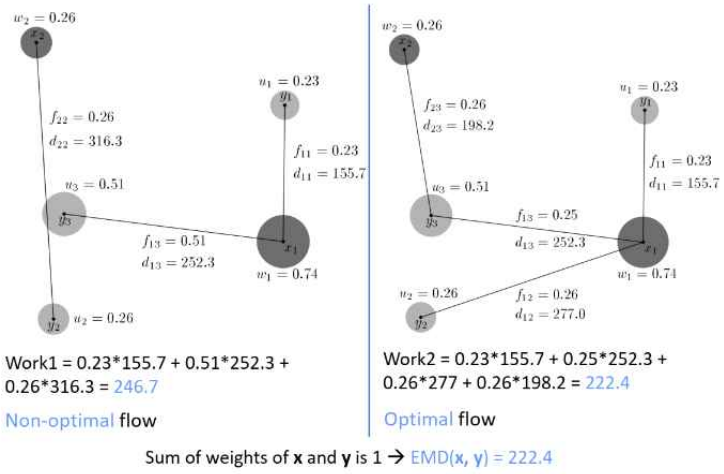
EMD는 이 work function의 최솟값을 의미합니다.(정확히는 최솟값을 $\sum_i \sum_j f_{ij}$로 나눈 값)

(요약하면, A의 vertex들에 있던 흙더미들($w(v_i)$만큼 각각의 vertex에 있음)이 B의 vertex들이 각각 $w(u_j)$의 흙더미들을 가지도록 옮겨지는데, 각 흙더미들이 이동하는 거리의 합(work function)을 최소로 만들 때 이 최솟값을 의미합니다.

(수업에서 EMD 는 각 element들이 bijection(전단사 함수)을 가져야 한다고 했는데, A의 vertex의 각 흙더미 알갱이들이 B의 어느 vertex로 갈지를 결정짓는 함수를 의미하고, 결국 이 모든 가능한 함수들 중 work function을 최소로 만드는 함수를 찾아 그 work function을 구하는 것을 의미합니다.)

=> super source s*, super sink t*을 잡고 s*과 A의 vertices들을, B의 vertices들과 t*을 연결시킨 후 minimum cost flow 문제를 푸는 식으로 접근해야 할 것 같습니다.

ex)



- Applying to Sphere Set

$$A = \{x_1, x_2, \ldots\} \, (x_i : center\ of\ sphere)$$

$$B = \{y_1, y_2, \ldots\} \, (y_i : center\ of\ sphere)$$

(Chamfer Distance와 동일하게 설정)

$$w(x_i) = w(y_j) = 1$$

(우리 문제의 경우 각 vertex(sphere center)의 weight은 모두 1으로 설정합니다.)

sphere set A, B의 집합 크기가 4^{level}으로 동일하기 때문에(*leaf level에서는 weight을 재분배시키지 않는 한, EMD를 적용시킬 수 없습니다.) EMD를 구할 수 있고, edge의 capacity도 1로 설정하면 위의 일반적인 경우보다 문제가 조금 단순해집니다.

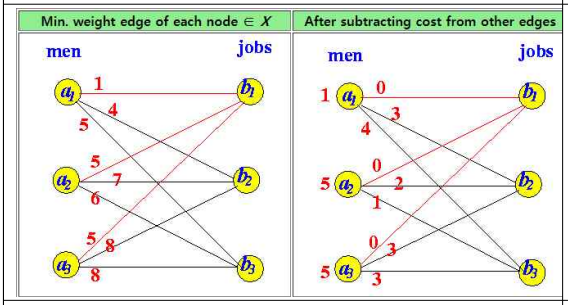
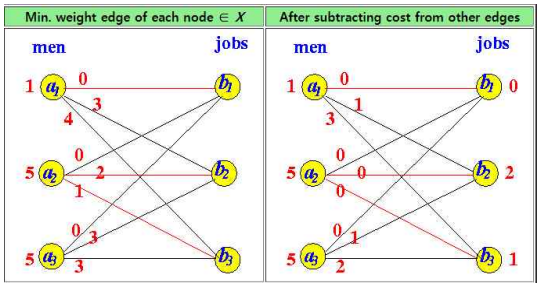
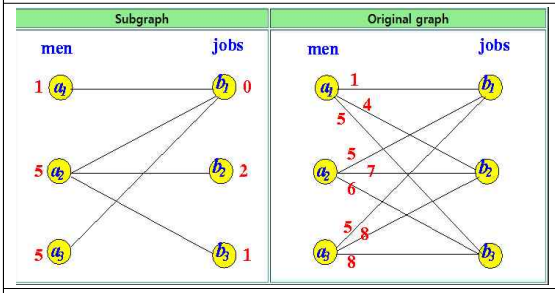
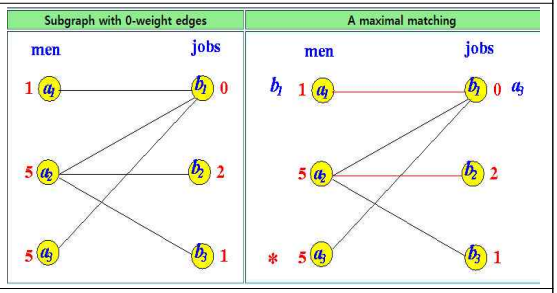
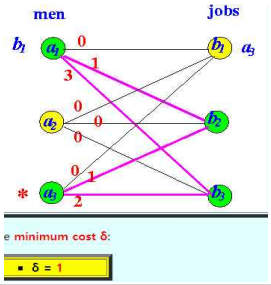
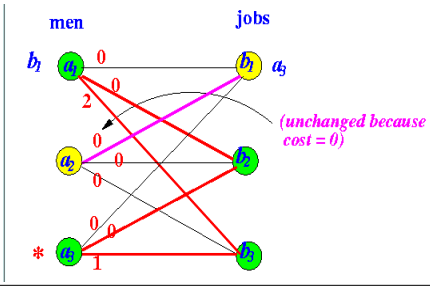
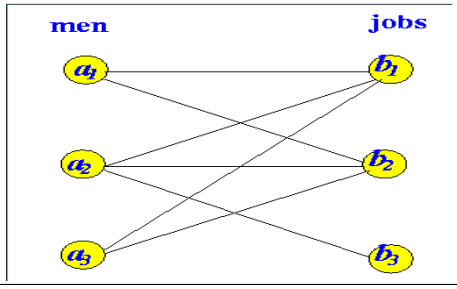
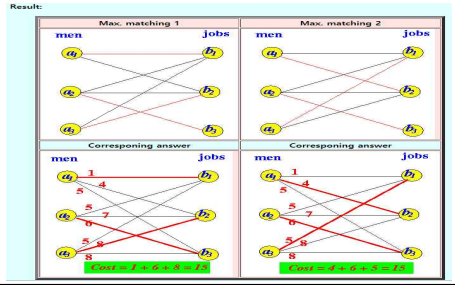
A의 vertex들과 B의 vertex들을 모두 연결시켰다고 생각하면 weight edge(weight은 각 sphere center 사이의 distance)가 있는 Bipartite graph에서 Minimum weight bipartite matching Problem이 됩니다.

1) 첫 번째 방법(Find Optimal)

찾아보니 Hungarian Algorithm이라는 algorithm이 이 문제와 완전히 같은 문제를 푸는 알고리즘이 있어서 다음은 Hungarian Algorithm에 대한 내용입니다.

```
//[Hungarian Algorithm]

EMD_optimal (sphere_tree A, sphere_tree B, level k){

//Initial Step 1
  for each sphere x in A in level k{
    d_min = min_distance(x, B);
    subtract d_min from all weights(or distances) between x and spheres in B.
    //We can get at leas one 0-weight(distance) edge for each node.
  }
  for each sphere y in B in level k{
    d_min = min_distance(y, A);
    subtract d_min from all weights(or distances) between y and spheres in A.
    //We may get more 0-weight(distance) edges
  }
//Initial Step 2
  //Now consider the subgraph consisting only of the 0-weight edges
  //These edges will provide the lowest possible cost if we can find a maximum matching

  find_maximal_matching(subgraph);// if we can find maximal matching, we are done.
  //when we are executing the find_maximal_matching function, we add the 'label' or 'flag' at vertices
which has matching.

//Iterative Step
//Add the "next least cost edges" to the subgraph, then find a new maximal find_maximal_matching

  while(matching is not maximal){
    //we are looking at the original bipartite graph(before Initial Step 2)
    minimum_cost = min{weights(distances) from labeled sphere in A to unlabeled sphere in B}

    for each edge with weight>0 such that from labeled sphere in A to unlabeled sphere in B{
      subtract minimum_cost from the cost of the edge
    }
    for each edge with weight>0 such that from unlabeled sphere in A to labeled sphere in B{
      add minimum_cost from the cost of the edge
    }

    add new 0-cost edges into subgraph from Initial Step 2
    find_maximal_matching(subgraph);
  }
}
```

(이해를 돕기 위한 단계별 그래프 figure들입니다.)

1	2
	
3	4
	
5	6
	
7	8(result)
	

2) 두 번째 방법(By Heuristic)

optimal matching을 찾기 너무 어려우니 optimal에 가까울 것으로 추정되는 bijection 함수 두 개의 workfunction 값의 평균을 사용하는 방법입니다.

(대강 비슷하게 나오지 않을까요..?)

함수1

-sphere_set A의 각각의 sphere_center에서 순서대로 가장 가까운 B의 vertex 하나씩 배정

함수2

-sphere_set B의 각각의 sphere_center에서 순서대로 가장 가까운 A의 vertex 하나씩 배정

(Chamfer Distance와 다른 것은 EMD의 경우 bijection이어야 하므로 A의 다른 두 개의 center v_i, v_j의 minimum distance vertex가 동일하게 u_k라고 하더라도 처음 배정한 v_i에만 u_k를 배정하고, v_j에는 아직 배정되지 않은 vertex중 minimum distance vertex들을 배정하는 방식)

```
EMD_heuristic (sphere_tree A, sphere_tree B, level k){
  EMD=0;
  Q_A=list of spheres in A
  Q_B=list of spheres in B
  for each sphere x in A in level k{
    cd+=minimum_distance(x,Q_B);
  }
  for each sphere y in B in level k{
    cd+=find_minimum_distance(y,Q_A);
  }
  return cd/2;
}

minimum_distance(sphere x, Q_B){
  min_d=0;
  for each sphere y in Q_B{
    min_d = min(min_d, distance(x,y));
    tmp=y;
  }
  pop out tmp from Q_B
  return min_d
}
```