ch.08 근사 문제

다항식 시간 내에 해결하지 못 하는 NP-완전 문제들에 대해서 최적해 대신 근사해를 찾음

근사 알고리즘

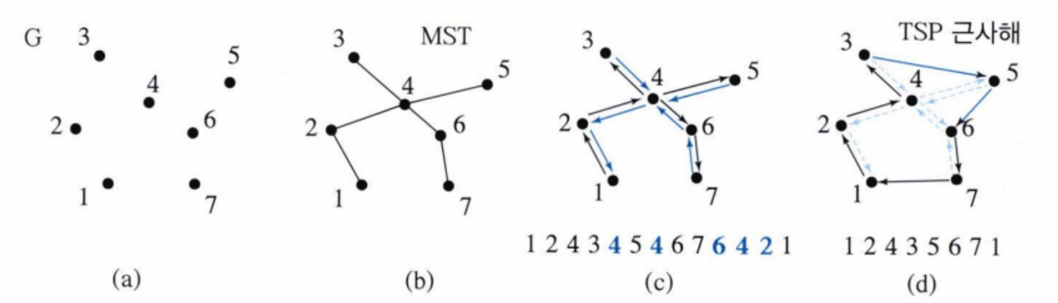
최적해에 가까운 해를 찾아줌 → 다항식 시간 복잡도

- 근사 비율(approximation ratio) : 근사해가 얼마나 최적해에 근사한 것인지를 나타냄
  - 1.0에 가까울수록 정확도가 높은 알고리즘
  - 간접적인 최적해를 찾아 근사 비율을 계산

8.1 여행자 문제(TSP)

여행자가 임의의 한 도시에서 출발하여 다른 모든 도시를 1번씩만 방문하고 다시 출발했던 도시로 돌아오는 여행 경로의 거리를 최소화하는 문제

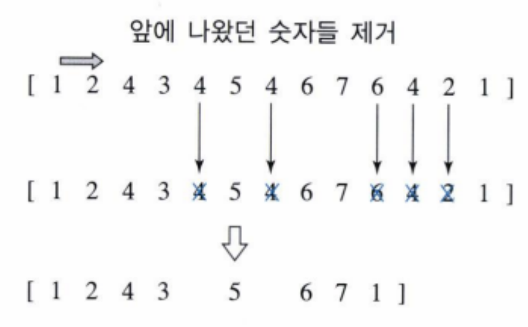
- 조건
  - ① 대칭성 : A→B = B→A
  - ② 삼각 부등식 특성 : A→ B < A→C→B



→ Krusckal or Prim

1. MST 찾기 : 모든 점을 사이클 없이 연결하는 트리 중 간선의 가중치 합이 최소인 트리
2. 시작도시를 제외한 다른 모든 도시를 트리 간선을 따라 1번씩 방문하도록 경로

3. 시작도시를 제외한 중복 방문하는 도시를 제거




```
Approx_MST_TSP()

입력 : n개의 도시, 각 도시 간의 거리
출력 : 출발 도시에서 각 도시 1번 씩만 방문, 출발 도시로 돌아오는 도시 순서 입력에 대한 MST

1. 입력에 대하여 MST 찾음 //prim or kruskcal
2. MST에서 임의의 도시로부터 출발하여 트리의 간선을 따라서 모든 도시를 방문하고 다시 출발했던
  도시로 돌아오는 도시 방문 순서 찾기
  //방문 순서가 정해져있지는 않아서 임의의 순서로 방문

3. return 이전 단계에서 찾은 도시 순서에서 중복되어 나타나는 도시 제거한 도시 순서
(가장 마지막의 출발 도시는 제거 x)
  //삼각부등식의 원리 적용 + 중복 방문된 도시 제거
```

Time Complexity

$O(n^2)$ $O(m\log m)$

① • MST : prim, krusckal ⇒ m간선수, n 정점

② • 방문 순서 찾기 : O(n) (간선 수가 n-1)

③ • 중복된 도시 제거 → O(n)

⇒ krusckal, prim 시간복잡도와 동일

Approximation Ratio

실질적인 최적해 알 수 없음 → 간접적인 최적해 사용

∴ 간접적인 최적해 : 최소 신장 트리 간선의 가중치의 합(M) < 실제 최적해 값

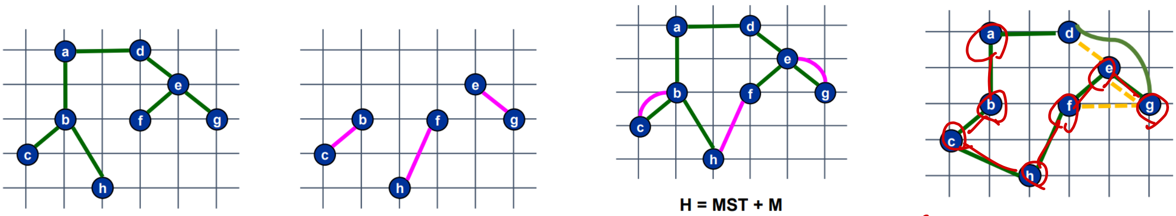
- ① 근사해 ≤ 2M
  - 도시 방문 순서를 찾을 때 각 간선이 2번 사용됨 → 경로 총 길이 2M
  - 삼각 부등식의 원리(a+b > c)를 이용 → 새로운 도시 방문 순서를 만들기에 경로가 더 짧음
- ② 근사 비율 ≤ 2 = 2M/M

+) 근사 비율 ≤ 1.5 → 추가 강의자료 볼 것

+) Christofides's Algorithm

1. T = minimun-spanning-tree(G)
2. O : T 안에서 홀수 개의 간선을 가진 정점 집합 → O는 짝수 개의 정점
3. M : O에 의한 부분 그래프에서의 minimun-weight-perfect-mahching
4. H : T+M → 짝수 개의 간선
5. H를 방문한 정점은 제외하면서 각 정점을 traverse

TSP 적용



MST T　　Matching M　　H = MST + M　　Avoid visited nodes

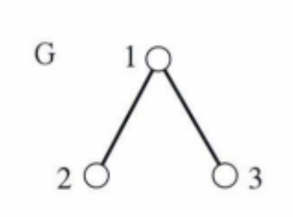
a-b-c-~~b~~-h-f-e-g-~~e~~-d-a

a-b-c-h-f-e-g-d-a

- approximation ratio
  - denote
    - OPT : optimal solution
    - A : 알고리즘에 의해 선택된 간선 그래프
    - EC : 오일러 circuit에 의한 간선 집합
  - cost(M) ≤ cost(OPT) / 2
    - 모든 부분집합 B에 대한 minimum tour ≤ cost(OPT)
    - 해밀턴 cycle을 수행한 B는 B에 속하지 않은 정점들은 skip → OPT보다 작음
    - B는 홀수번째 간선만 선택한 B1, 짝수번째 간선만 선택한 B2를 생각할 수 있음
    - cost(M) ≤ min(cost(B1), cost(B2)) ≤ cost(B)/2 ≤ cost(T)
    - 모든 정점이 짝수개의 간선을 가짐 → 대응 가능
  - 오일러 사이클을 수행한 EC는 (M U T)에 속함
  - cost(EC) = cost(T) + cost(M) ≤ 1.5cost(OPT)
  - cost(A) ≤ cost(EC) ≤ 1.5cost(OPT)

8.2 정점 커버 문제(Vertex cover)

주어진 그래프 G=(V,E)에서 각 간선의 양 끝점들 중에서 적어도 하나의 끝점을 포함하는 점들의 집합들 중에서 최소 크기의 집합을 찾는 문제

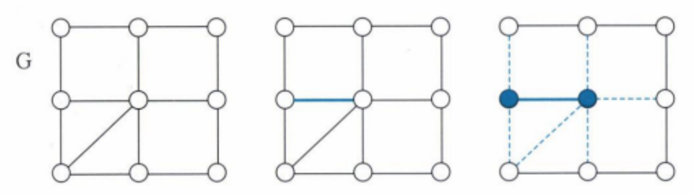


ex) 모든 복도(간선)을 ‘커버’하기 위해 CCTV(정점) 카메라의 수와 위치를 찾는 것

- **정점 선택 방법**
  1. 차수(degree)가 가장 높은 점을 우선 택(간선이 많은 점) → 많은 수의 간선이 커버
     a. set cover의 근사 알고리즘 → 근사비율 : logn
  2. 간선을 선택 → 선택된 간선의 양 끝 점에 인접한 간선이 모두 커버 → Vertex cover

  ⇒ 2번 사용

- **vertex cover algorithm**
  1. 간선을 선택
  2. 새로 선택할 간선 : 양 끝점이 이미 선택된 간선의 양 끝점 집합에 포함되지 않으면 선택



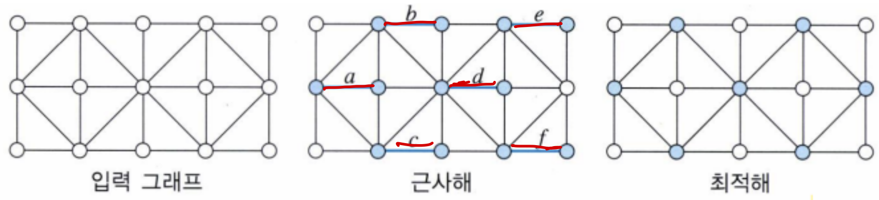
  3. 더 이상 간선을 추가할 수 없을 때 중단
     → 이때의 간선 집합 = 극대 매칭(amximal matching) : 간선의 양 끝점 중복 x
     → 극대 매칭을 활용하여 정점 커버 근사 알고리즘의 근사해를 구함

```
Approx_Matching_VC(){
입력 : 그래프 G=(V,E)
출력 : vertex cover

1. 입력 그래프에서 극대 매칭 M을 찾는다
```

↳ ① 간선을 양 끝점 중복되지 않도록 선택

```
retrun M의 간선의 양 끝점들의 집합
```

- ex)



Time Complexity

- 극대 매칭 시간 복잡도와 동일
    - 하나의 간선 e 선택 → e의 양 끝점과 인접한 모든 간선을 제거 → O(n)
    - 간선 개수 : m

⇒m * O(n) = O(mn)

Approximation Ratio

실질적인 최적해 알 수 없음 → 간접적인 최적해 사용

∴ 간접적인 최적해 : 극대 매칭(M)에 있는 간선 수

- 근사해 = 2M
    - 각 간선의 양 끝점들의 집합을 정점 커버의 근사해로서 return → 극대 매칭 간선 수 * 2
- 근사 비율 = (극대 매칭의 간선 양 끝점 수) / (극대 매칭의 간선 수) = 2

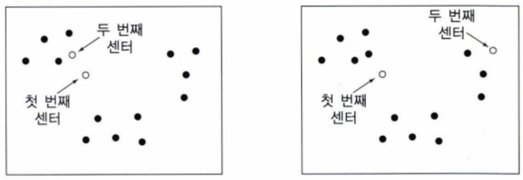
8.5 클러스터링 문제(Clustering)

n개의 점이 2차원 평면에 주어질 때, 이 점들 간의 거리를 고려하여 k개의 그룹으로 나눔

각 그룹의 중심이 되는 k개의 점을 가장 큰 반경을 가진 그룹의 직경이 최소가 되도록 선택
⇒ 그리디 알고리즘

- 센터를 정할 때, 정해진 센터들과 가장 멀리 떨어진 점을 센터로 선택 → 직경은 최소

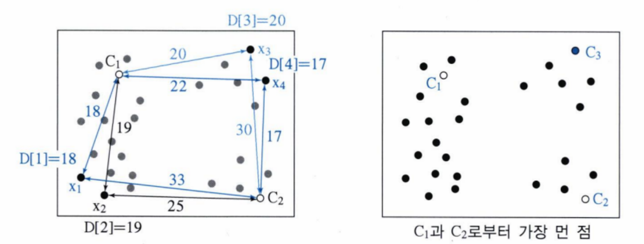
한꺼번에 선택보다는
한 번에 하나씩 선택이
더 쉬움!



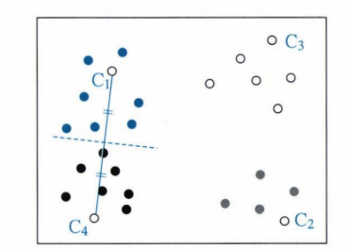
첫 번째 센터에서 가장 가까운 점 / 첫 번째 센터에서 가장 먼 점

```
Approx_k_Clusters()

입력 : 2차 평면상의 n개의 점 xi (i=0 ~ n-1, 그룹의 수 k > 1)
출력 : k개의 클러스터 및 각 클러스터의 센터

c[1] = r //단 xr은 n개의 점 중에서 랜덤하게 선택된 점

for j=2 to k{
  for i=0 to n-1{
    if(xi != 센터)
      xi와 각 센터까지의 거리를 계산하여, xi와 가장 가까운 센터까지의 거리를 D[i]에 저장
  }
  C[j] = i //단, i는 배열 D의 가장 큰 원소의 인덱스, xi != 센터
}

센터가 아닌 각 점 xi로부터 k개의 센터까지의 거리를 각각 계산
그 중 가장 짧은 거리의 센터를 찾음
해당 센터의 클러스터에 xi 속함

return 배열 C, 각 클러스터에 속한 점 리스트
```

① C_1 결정
② 각 점에 대해
x_i에서 C_1까지의 거리
↳ D[i]
③ 가장 거리가 먼 점 ⇒ C_2
④ x_i에서 C_1까지의 거리
x_i에서 C_2까지의 거리
↳ 계산하여 작은값을 D[i]
⑤ 가장 거리가 먼 점 → C_3
⋮
C_4까지 정함
⇓
센터가 아닌 점들 모두
가장 가까운 센터를 찾아서
그 클러스터에 속함.



(a) / C1과 C2로부터 가장 먼 점 (b)



time complexity

- 임의의 점을 선택 → O(1)
- for - loop : (k-1)번
  - 각 점에서 각 센터까지의 거리 계산 → O(kn)
  - 최댓값 찾기 → O(n)

  → (k-1) *(O(kn)+O(n))
- 센터가 아닌 각 점으로부터 k개의 센터까지의 거리를 각각 계산하여 최솟값 찾음 → O(kn)

⇒ O(1) + (k-1)(O(kn) + O(n)) + O(kn) = O(k^2n)

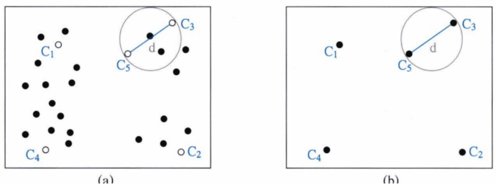
Approximation Ratio

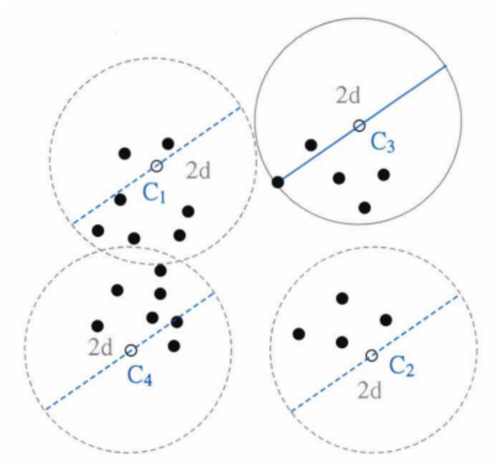
- OPT : 가장 큰 직경
    - k개의 센터를 모두 찾고, (k+1)번째 센터를 찾는다고 가정
    - k=4 → 4개의 클러스터로 분할해야 함
    - 5(k+1)개의 센터 중 2개는 하나의 클러스터에 속함

      → 둘의 거리인 d보다는 직경이 커야 함

  ⇒ OPT ≥ d

  ↳ 거리가 가장 먼 점을 선택





- OPT' :가장 큰 그룹의 직경 ≤ 2d
- 2OPT ≥ 2d ≥ OPT'

  ⇒ 근사 비율 : 2를 넘지 않음

+) k-means Algorithm ⇒ EM (기대값 최대화 알고리즘) → 비용 함수의 극대점으로 단조 수렴

① cluster 개수 k 설정

② 초기 중심점 설정 → 매우 중요! (OPT 달라짐) ⇒ 여러가지 선택으로 시도해보아야 함.

↳ 교재에서 소개된 방법 : Random

↳ 최적해 : $\vec{\mu}(c) = \frac{1}{|c|}\sum_{x \in c}\vec{x}$

③ 데이터를 cluster에 할당

④ 중심점 재설정

↳ 교재 : data의 평균 지점 선정

⑤ 다시 cluster에 데이터 할당

⇒ 정해진 반복 횟수만큼 ④,⑤ 반복!

* 조건

① 반복 횟수는 fixed

② Point partition은 바뀌지 x

③ Centroid position 〃

* Time Complexity

① 거리 계산 → O(M)

② cluster 재설정 → O(kNM)

③ 중심점 계산 → O(NM)

④ 반복 횟수 I 일 때

⇒ O(IkNM)