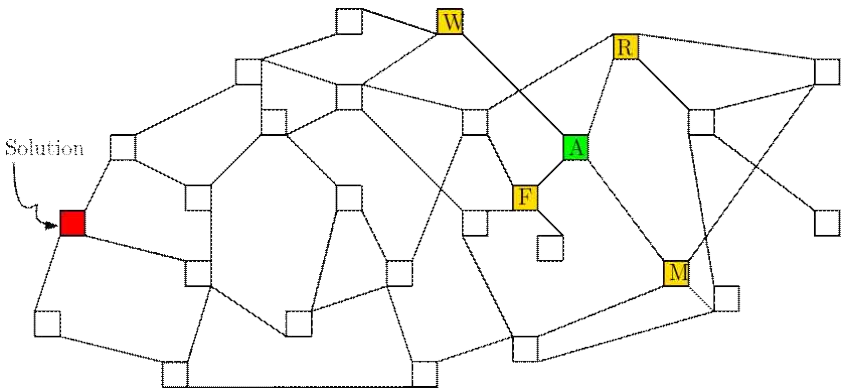
Lecture 23 : Branch and Bound Algorithm의 기초

Branch and Bound(이하 BB)방법은 휴리스틱 알고리즘의 골격을 이루는 방법론이며 인공지능(AI) 분야에서 A*라고 불리는 방법과도 본질적으로 동일하다. 이 방법을 간결히 요약하면 앞서 설명한 되추적(backtracking) 방법의 개선판(improved version)이라고 할 수 있다.

대부분의 계산 문제, 알고리즘 문제는 state space로 탐색공간에서 하나의 상태를 찾는 문제로 환원(reduce)할 수 있다. state이란 문제 풀이 과정의 특정한 한 상태를 나타내는 것으로 이것을 모두 모아둔 집합이 전체 탐색 공간이다. 그런데 이 state들 사이에는 "인접한 관계"가 설정되어 있는데 하나의 기본동작을 통해서 상태 s_a가 상태 s_b로 바뀌면 이 둘을 edge (s_a, s_b)로 연결한다. 이와 같은 작업을 거치면 탐색공간은 결국 하나의 그래프(graph)로 변화도게 된다. 따라서 모든 계산 문제는 결국 크기와 내용을 알 수 없는 미지의 그래프 상에서 한 특정 노드, 즉 정답 노드를 찾아나가는 과정이라고도 볼 수 있다. 문제는 이 정답 노드를 얼마나 빠르게, 그리고 최소의 비용으로 찾아나갈 수 있는가-인데 이 성능에 따라서 좋은 알고리즘, 나쁜 알고리즘, 최적(optimal) 알고리즘이 나타나게 된다. 아래 state space로 표현된 그래프를 보면서 이야기를 이어 나가보자.



우리는 지금 A 상태에 있으며 전체 그래프가 어떻게 구성되어 있는지 모르는 상태이다. 즉 노드의 수도 모르고 각각의 state가 어떻게 연결되어 있는지도 모른다. 문제는 이 상황에서 A에서 출발하여 목적지, 붉은색 노드(solution node)를 찾아가야 하는 것이다.

혹자는 state space 그래프 전체를 미리 완전한 모양으로 구성한 다음에 A에서 Solution으로 가는 Shortest path를 구하면 되지 않냐고 주장할 수 있지만 문제에 따라서는 전체 공간의 그래프를 만드는 것 자체에 불가능하다. 예를 들어 n개의 원소로 구성된 bin packing 문제의 경우 가능한 state의 수는 2^n[1]가 되므로 n=100만 되어

1) 각 항목은 bin이 들어가는 경우와 들어가지 않는 경우, 이 두 가지 경우가 가능하므로 bin에 원소가 담길 수 있는

서 전체 공간의 수는 대략 10^{30} 정도가 되므로 현실적으로 이 모두를 다 살펴보는 것은 불가능하다. 또 경우에 따라서는 위 그림에서 붉은 점으로 표시한 solution node를 알 수 없는 경우도 발생한다. 따라서 현실적으로는 위 상태에서 시작점 A에서 다음으로 진행되는 가능한 경우의 수, 인접한 노드 **{W, R, F, M}**중에서 하나로 움직여야 한다. 결국 탐색 알고리즘이란 이 4개의 경우 중에서 어떤 노드로 가는 것이 가장 빠른 정답으로의 길을 보장하는가의 문제로 귀착된다.

BB의 방법은 이 과정의 일부이다. 일단 A에서 가능한 4개의 노드로 살짝 나가본다. 이 과정이 바로 확장(branch)과정이다. 이 4개 중에서 가장 유망한(promising) 것 하나를 찾는다. 그리고 다시 그 노드에서 branch를 수행한다. 이렇게 진행하면 우리는 가능성이 있는 노드(제일 front)들만을 관리하게 된다. 그 관리되고 있는 노드 중에서 가장 가능성있는 경우를 먼저 뽑아서 이로부터 탐색공간을 전개(branch)해야 하므로 이를 하기 위해서는 자료구조에서 배운 Priority Queue가 필수적이다. 이 BB 방법은 핵심기술은 Bounding 기술이다. 만일 관리하고 있는 노드 중에서 현재 확보된 값보다, 지금도 열등하고, 앞으로도 계속 열등함이 확인된다면 이 노드를 더 이상 고려 대상이 되지 않고 바로 “처단”된다.

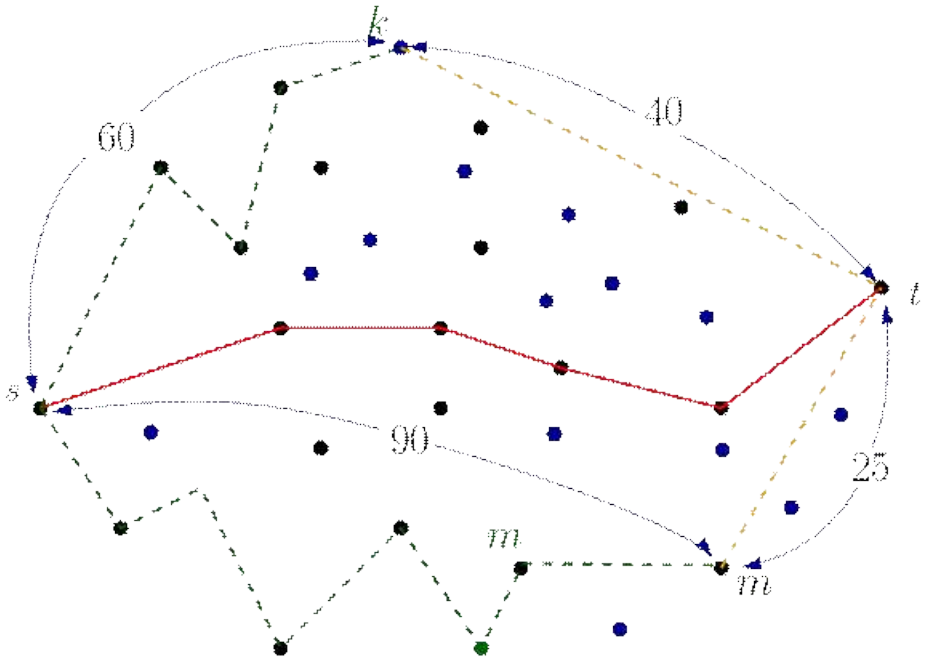
좀 억지스럽지만 젊은이들의 생활에 맞추어 비유적으로 설명하면 다음과 같다. 어떤 사람이 결혼 또는 연애를 목적으로 다수의 이성을 사귀고 있다. 이 중에서 다음 주말의 데이터 상대는 가장 promising한 사람이 선택된다. priority queue를 통하여 뽑는다. 물론 어장에 “관리”중인 다른 사람에게도 일정한 관심을 줘서 도망가지 않도록 관리한다. 만일 어떤 데이터를 하던 중 특정인이 보여주는 정성, 앞으로의 현실적인 기대값이 현재 관리되고 있는 사람 중에서 내가 확보한 사람보다 못함이 확인되면 이 사람은 바로 퇴출된다. 즉 이 사람이 최종의 후보로 선출된 가능성은 없기 때문이다. 이 과정을 최고의 정답을 찾을 때까지 반복하는 것이 BB이다. 즉 BB가 끝나는 경우는 다음과 같은 경우이다.

i) state space를 모두 빠짐없이 탐색이 된 경우
ii) 하나의 답을 찾고 나머지 노드는 모두 bound 되어 더 이상 가망성(promising)이 있는 노드가 없는 경우.

다른 예를 들어 설명해보자. 어떤 도로망에서 s에서 출발하여 t까지 가는 최단거리를 계산하려고 한다. 이 과정에서 중간 노드를 거쳐 가는데 단 이동하는 두 점 (x,y)의 거리는 최대 L을 넘어갈 수 없다. 이 조건이 없으며 s에서 t로 바로 날아가는 직선 경로가 최소가 되기 때문이다. [2] 아래 그림에서 우리는 s에서 목적지 t로 가는 경로 72짜리 path(shortest)를 찾았다고 가정한다. 이 상황에서 노드 k와 m는 bound가 된다. 왜냐하면 s에서 k까지의 거리가 60, m까지의 거리가 90이다. 그런데 k에서 t까지 직선으로 가는 경로만으로도 40이므로 s에서 k까지 온 길로는 현재

모든 경우는 n개 원소로 이루어진 집합의 power set이 되어 그 갯수는 2^n이 된다.
2) edge length constraint shortest path problem이다.

확보된 최단거리 72보다 절대 더 짧을 수가 없다. 따라서 이 길은 더 이상 고려되지 못하고 bound가 된다. 파란색 dashed line으로 표시된 경로 역시 같은 방식으로 bound된다.



23.1 Backtracking(BT)과 Branch and Bound(BB)의 비교

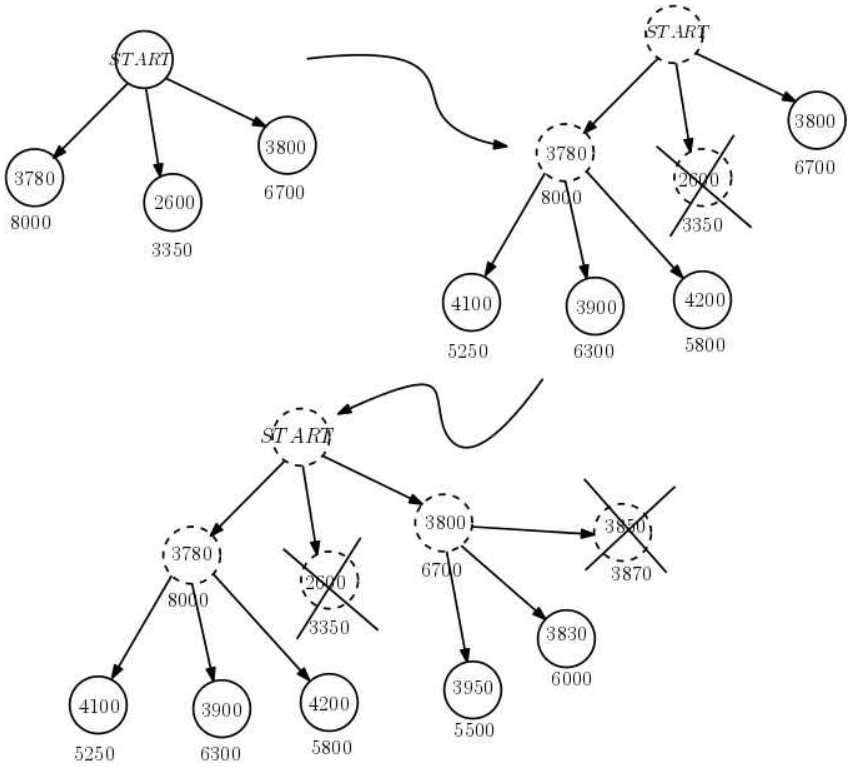
- Backtracking의 방문순서는 기계적, Solution의 상태와 무관
- Searching space를 어떻게 관리하는가? { Total Ordering, 저장}
- 탐색 상태를 저장하는 방법 { stack, queue }
- 계산을 많이 활용할 것인가 아니면 이미 계산하여 저장된 값, 즉 메모리를 많이 활용할 것인가, 또는 그 균형을 잡을 것인지를 결정

세상의 모든 방법 = Data Approach, Algorithm Approach
기본적인 자료구조는 Priority Queue vs Stack(Backtracking)

23.2 B&B 알고리즘의 철학:

규칙 1: 가능성을 모두 열어둔다.
규칙 2: 그 중에서 제일 유망(promising)한 것부터 고려해본다.
규칙 3: 이미 확보된 이익보다 전망이 나쁜 미래일정은 버린다. 이것이 B&B방법의 성능을 결정하는 가장 중요한 기술이다. 가능성이 없는 일은 더 이상 쳐다보지 않는 것은 일상에서도 꼭 필요한 삶의 지혜와도 비슷한 의미라고 할 수 있다. 3)

3) 러시아 속담에 이런 말이 있다. "공부와 사랑은 할 수 있을 때 해야 한다. 하고 싶을 때가 아니라". 너무 늦게 일상의 내버리고 자신만의 취미생활에 몰두하는 것은 절대 바람직한 삶이 아니다. 가능성이 0.01%도 되지 않는 아이돌 스타를 꿈꾸며 학원이나 기획사를 믿는 것을 용기라고 북돋우어줘서는 곤란하다.



Q) 좋은 직장을 구하는 과정을 보장된 연봉, 연봉 상한, 복지 등의 부가조건을 활용하여 B&B 방법을 비유적으로 설명해보자.

23.3 알고리즘 성능을 개선해주는 일반적인 방법론(methodology)

a. 이미 한 계산을 두 번 다시 하지 않는다.
b. 다음에 쓸 가능성이 있는 결과를 최대한 기억시켜서 재활용한다.
c. 한 번의 계산에서 얻을 수 있는 모든 정보를 최대한 뽑아낸다.
(예를 들면 second max, max+min 문제와 같이.)
d. 답이 될 가능성이 있는 것부터 먼저 살펴본다. (하나의 답을 찾는다면)
e. 답이 될 가능성이 없는 것은 일찍 잘라낸다(pruning).

23.4 Branch and Bound 알고리즘 설계방법론의 주요 접근법, 아이디어

a. 문제에서 정해진 고정된 순서가 아니라
b. 현재 상황에서 답이 될 가능성이 있는 노드부터 탐색을 한다.
c. 단 이미 방문한 노드와 아직 방문하지 않은 것, 그리고
앞으로 방문할 예정인 것을 명확하게 구분하여
기억하고 있어야 한다. 비록 상당한 비용이 들더라도
d. 따라서 필요한 자료구조는 Priority Queue를 사용한다.
(Heap.. 우선순위 큐... 특정한 값이 순서가 partially 정리된,,,)

즉, 깊이 중심이 아니라 답이 있는 것부터
답이 될 가능성의 "냄새"를 맡아가면서 탐색공간을 조금씩 확대한다.
(단 이 과정에서 그 순서를 저장하고 있어야 하는 자료구조관리가
부가적으로 필요하다.)

e. 이 방법 역시 exhaustive searching의 한 가지이지만 Backtracking에 비해서 조금 더 수행시간을 개선(reducing)할 수 있는 대안이 된다.

f. 단 보조 자료구조를 추가로 준비해야 하므로 공간 복잡도는 증가한다.

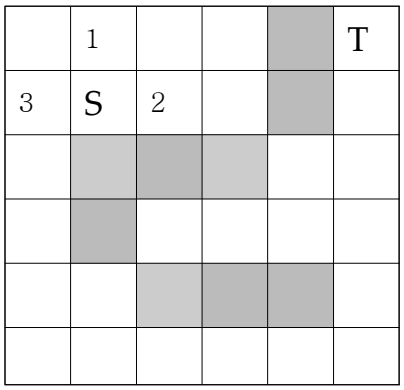
23.5 Q) 마음에 드는 옷을 백화점에 가서 구입하려고 한다. 백화점을 Backtracking으로 뒤져가는 방법과 Branch and Bound 방식으로 찾아가는 것을 비유해서 설명해 보시오.

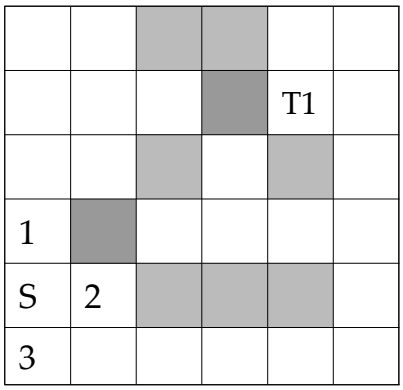
Q) 어떤 문제에는 Branch and Bound가, 어떤 문제에는 Backtrack 방법이 더 좋은지 판별하는 기준이 있을지 예를 들어 설명해보자.

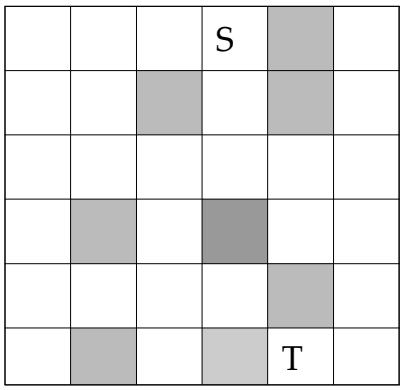
23.6 파리(Paris)에 배낭여행을 갔다. 현재 위치 S에서 볼 때 에펠탑 T가 살짝 보인다. 여러분이라면 어떻게 가장 짧은 길을 찾아갈 것인가? 만일 앞에 장애물이 있으면 앞으로 목적지 예상 거리에 +3씩 추가의 벌점(가점)을 부여한다.

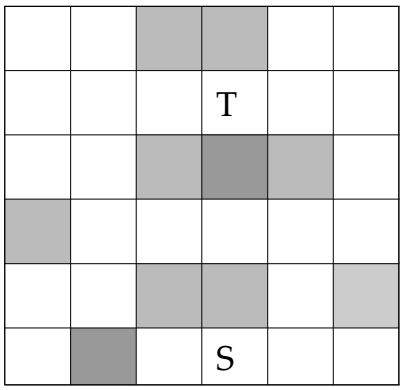
- 현재 "active"된 노드에서 Branch를 진행한다.
- 현재 Branch 된 노드 중에서 가장 "똘똘한 놈" 하나를 잡아 처리한다.
- 다시 자손 중에서 가장 똘똘한 놈을 찾아서 이 작업을 계속한다.
- 예) 맨해튼(Manhattan)[4]에서 길 찾아가기
  S에서 T까지 가는 길을 "빨리" 찾으시오.

4) 모든 도로가 수직 수평 성분뿐이므로 도로 역시 수직 수평길로만 이루어져 있다. 따라서 맨하탄 거리로 볼 때 (0,0)과 (1,1)까지의 "직선거리"는 $\sqrt{2}$ 가 아니라 1+1=2로 계산된다.





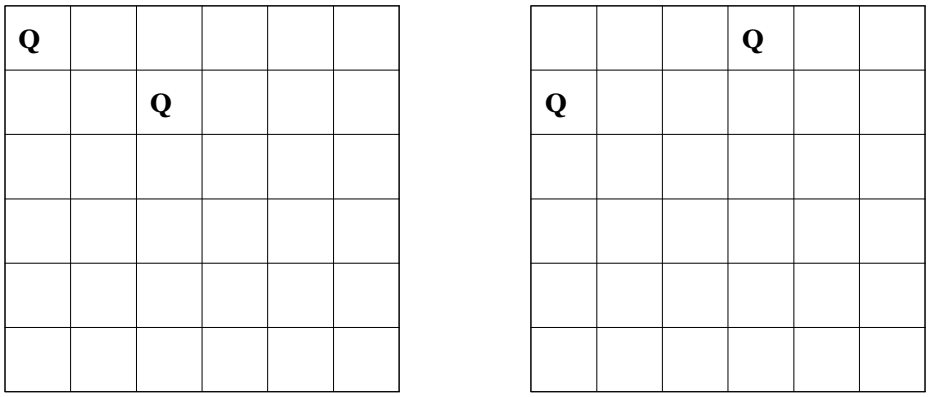




Q) 이 문제를 backtrack으로도 한번 시도해 봅시다. backtrack에서 찾아가는 방향의 윗부분에서 시작하여 시계방향으로 돌면서 진행한다.

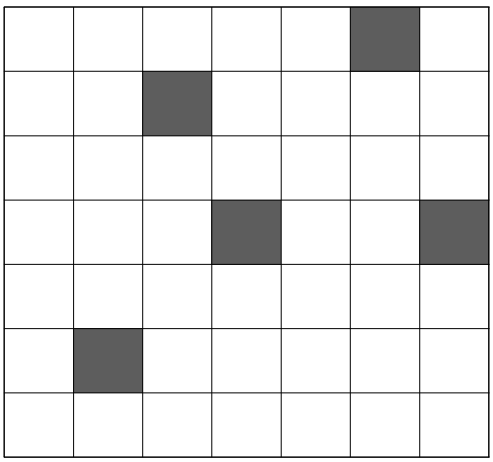
Lecture 24 : Branch and Bound 알고리즘의 실전 응용

24.1 6-Queen Problem (Queen은 가로, 세로 대각선으로 공격할 수 있다. 6개의 Queen이 모두 안정된 위치에 있도록 배치하는 방법. 가장 가능성이 있는 곳을 골라서 먼저 놓아본다. (사실 이 문제는 간단한 linear time solution이 있다. 그렇지만 그것을 모른다고 가정하고 풀어 봅시다.)



중요사항: Branch and Bound는 Backtrack에 비해서 평균적으로 빠를 뿐
- Order로 본다면 별 차이가 없다.
- Worst Case로 본다면 전혀 개선되지 않은 방법이다.
- 앞서의 에펠탑 찾기 문제에서 가장 최악인 경우의 지도를 표시하시오.

24.2 이전과 같이 무식하게 Backtracking으로 푼 n-Queen Problem의 좀 멋있게 풀어 봅시다. i번째 줄에서 다음 i+1 번째 줄로 내려갈 때, 정해진 순서(예를 들면 왼쪽부터 오른쪽)대로 하지 않고 가장 “가능성”이 있는 길부터 먼저 탐색해보자. 다음은 n-Queen Problem인데 각 검은색 cell은 장애물을 나타낸다. 이것은 Queen의 공격을 막아준다. 이 문제를 Branch and Bound로 풀어 보자.



24.3 Branch and Bound를 이용하면 0/1 Knapsack을 가장 적절한 시간에 풀 수 있다. 즉 현재까지 확보한 이득(배낭에 들어있는 이득)과 배낭의 남은 공간에 채워 넣을 수 있는 이득의 예상치(물건을 임의의 비율만큼 잘라서 넣은 것이 허용되는 partial knapsack)으로 넣었을 때 확보되는 이득의 한계치를 구하면 된다.

- 현재까지 확보된 이득 (봉급) A
- 앞으로 가능할 것으로 예상되는 이득의 최대치 - B
- (A, B)의 쌍 모든 가능한 (A, B)값을 표시한다.

24.4 다음 2개의 사례를 사용하여 Knapsack 문제 (W= 55, W=30)를 풀어 보자. 단 이 문제에서 가성비는 이득/부피의 값으로 결정한다.

물건	부피	이득	이득/부피
C1	40	40	1
C2	20	35	1.5
C3	10	20	2
C4	15	45	3
C5	7	28	4
C6	12	60	5

물건	부피	이득	이득/부피
C1	20	40	2.0
C2	15	28	1.8
C3	10	27	2.7
C4	25	55	2.2
C5	8	28	3.5
C6	12	50	4.1

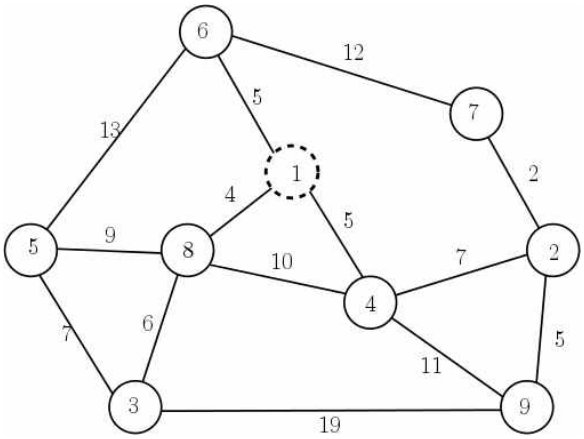
24.5 Backtracking을 Branch and Bound로 풀어 보자.
다이어트 식사를 위한 재료 장보기: 음식 재료를 선택하여 (단백질, 지방, 탄수화물, 비타민)=(100, 100, 100, 10)이상이 되도록 하는 방법. 단 필요한 영양을 확보할 수 있는 재료의 비용이 최소가 되어야 한다.

재료	단백질	지방	탄수화물	비타민	가격/W
1	30	55	10	8	100
2	60	10	10	2	70
3	10	80	50	0	50
4	40	30	30	8	60
5	60	10	70	2	120
6	20	70	50	4	40

24.6 Traveling Sales Person Problem.

어떤 판매원이 도시의 모든 지점(vertex)을 방문하면서 어떤 작업을 한다. 예를 들어 각 지점마다 그 주에 판매한 대금을 수금하여 온다든지, 정수기 filter를 바꿔어 준다든지 등이다. 목적은 모든 지역을 방문한 후 다시 본사로 돌아오는 길(path)의 경로가 가장 짧도록 출장계획 (방문할 vertex의 순서)를 정하는 것이다. 어떤 경우에는 불가피하게 한 지점을 1회 이상 방문하는 경우도 있다. 이 문제를 BT와 BB로 각각 해결하고 이 둘 방법을 사용했을 때 최종 결과값이 동일한지를 확인해 보고 두 방식의 효율성을 노드를 방문한 횟수 기준으로 비교하여 보시오. 두 방법 모두 출발은 1번 노드에서 시작한다.

a) 일단 1번에서 출발한다. 선택할 수 있는 경우는 { 4, 6, 8 }



b) 이 상황까지를 tree를 그려보자. 이게 leaf node {8, 6, 4 }에서 고려할 수 있는 cycle의 예상 길이와 그 지점까지의 거리 {4, 5, 5}를 더해서 가장 작은 쪽을 선택해서 branch 한다.

c) 위 그래프에서 TSP 거리의 lower bound와 upper bound를 구해보자.

$$Lower \leq Opt \leq Upper$$

위 조건을 만족하는 Lower와 Upper를 계산해봅시다. 만일 Lower = 1, Upper=10000은 trivial bound이다. 우리는 Lower는 최대로 높여야 하고, Upper는 최대한 낮추어야 한다.

d) 만일 Upper bound가 100임을 알고 있다고 하자. 즉 이 문제의 정답은 100 이하임을 확인할 수 있다. 그런데 현재 solution이 120이라면 이 이하로의 branch 더 이상 고려할 필요가 없다.

24.7 BB를 실전에서 사용할 때 현실적인 대책과 주의해야 할 점.

a) 일단 가장 단순한 Greedy 방법을 사용해서 대략의 성능을 본다.

b) Dynamic Programming의 가능성이 있는지 생각해본다.

- Space Complexity가 너무 크지 않은지를 생각해본다.

c) 최적을 위해서 Branch and Bound 방법을 사용해본다.

- 시간을 대략 특정해본다.
- 데이터 크기를 $N, 2N, 4N, \ldots 2^kN$ 과 같이 이전 데이터에 비하여 그 크기를 2배로 늘여가며 추가로 변화된 성능을 살펴본다.[5] 데이터의 크기가 2배로 늘어날 때 시간과 computing resource의 비율도 이와 같이 2배씩 늘어나는지 확인해봐야 한다. 이것은 알고리즘 복잡도의 실제적인 차수(Practical asymptotic order)에 대한 중요한 감을 전해준다.

5) 이 방법은 매우 중요한 현장 기술이다.