Lecture 17 : Greedy Algorithm 모형을 적용할 때 주의사항

이번 장에서는 욕심쟁이법을 사용할 때의 주의사항에 대하여 다룬다. 가장 흔한 실수는 욕심쟁이법으로 풀면 최적의 답을 찾을 수 있다고 믿는 것이다. 컴퓨터 과학에 이런 오류는 제법 나타나나. 욕심쟁이법으로 풀면 의심없이 최적의 해를 찾을 수 있을 것이라는 믿음에 반하는 반례 (counter example)이 한참 뒤에 발견되는 경우가 있다. 가장 유명한 사건은 greedy triangulation으로 optimal triangulation을 달성할 수 있다고 많은 계산기하학 연구자들의 믿음과 달리 반례가 한참 뒤에 발견되었다.

만일 어떤 문제를 Greedy로 풀 때는 반드시 그것이 optimal을 보장하는지를 아주 야무진 (rigorous) 방법으로 증명을 해야만 한다는 것이다. 직관적인 믿음만으로는 크게 부족하다. 아래는 greedy로 최적의 답을 찾을 수 없는 대표적인 문제 중 하나인 optimal assignment 문제이다. Greedy algorithm이 최적의 보장하는 것은 문제의 구조가 matroid로 구성되어 경우인데, 이 내용은 꽤 이론적인 부분이라 본 강의에서는 생략한다. 관심있는 사람은 아래 각주의 내용을 참조하기 바란다. [1)]

17.1 최적의 미팅 Pair 찾기 : Optimal Assignment Problem

5명의 총각과 5명의 낭자가 단체 미팅을 통하여 짝을 찾고자 한다. 단 주선자는 가능하면 전체 참가자의 **만족도의 합**이 가장 높도록 짝을 만들고 싶어 한다. 어떻게 짝을 찾아주면 좋은지 생각해보자. 즉 목적함수(objective function)은 matching weight의 총합이다.

a) 이 문제를 Greedy Solution으로 풀어보자. 알고리즘은 다음과 같다.
   unmatched된 모든 쌍 중에서 가장 matching 값이 큰 쌍을 구한다.
   이를 matched pair로 넣고, 이후 고려할 대상에서 제외한다.
   위 작업을 모든 쌍이 선택되어 아무도 남아있지 않을 때까지 한다.

b) 위 a)에서 제시한 그리디 방법보다 더 나은 답(solution)이 있는지 찾아보자.

	W1	W2	W3	W4	W5
m1	5	5	6	8	9
m2	4	5	7	6	10
m3	7	3	5	7	9
m4	10	2	10	9	10
m5	9	10	8	9	5

1) https://jeremykun.com/2014/08/26/when-greedy-algorithms-are-perfect-the-matroid/

17.2 남자가 여자를 선택하는 경우와 여성이 남자를 greedy하게 선택하는 경우에 그 합이 서로 일치하는지를 알아보자.

17.3 그리디(greedy)하게 선택하면 최적이 아닌 경우를 체계적으로 만들어 내는 방법을 생각해보자. 문제는 아주 유명한 Optimal Assignment 문제로서 $O(N^2)$에 최적을 답을 찾을 수 있는 헝가리 연구자 알고리즘(Hungarian algorithm)으로 해결하면 된다. 2)그래프 이론의 주요 문제다.

17.4 그리디(greedy) 알고리즘으로 bin packing을 푼다. 이 경우 가장 크게 손해가 나는 case를 만들고 그 때의 bound가 lower bound임을 보이시오.

그리디 알고리즘은 constructive algorithm의 전형이다. 즉 전체 solution은 하나의 partial solution을 점차적으로 개선하여 완성하는데 이미 구성된 부분 solution는 다시 수정하지 않는다. 만일 이것을 허용하면 반복 개선 (iterative improvement) 알고리즘이 된다. 사실 이 둘에는 경계가 없으면 정도의 차이가 있다. 즉 어느 정도까지 수정을 허용할 것인지에 따라서 결정된다.

17.5 Bin Packing에서 이미 pack에 들어간 원소 중에서 최대 1개까지는 꺼내서 바꿀 수 있다고 가정해보자. 이 경우 다음의 case에 대하여 1-회 교환을 허용하는 최적의 greedy algorithm을 구현해보자.

입력1 C= 50		입력 2 C= 60		입력 3 C=100	
23		23		23	
33		33		33	
45		45		45	
27		27		27	
13		13		13	
60		60		60	
53		53		53	
9		9		9	

2) 이 문제는 그래프 알고리즘의 핵심 알고리즘 중 하나이다. 증명은 상당히 긴 과정이 필요하다.

Lecture 18 : 일정 계획(Job Scheduling)문제와 그 변형

Job Scheduling은 산업공학 등 수리과학을 다른 분야에서 따로 독립된 과목으로 강의가 진행될 정도로 다양하고 중요한 문제이다. 이렇게 복잡한 이유는 일정을 결정하는데 다양한 조건이 있기 때문이다. 그리고 이 일정 계획은 현실에서 가장 흔하게 나타나는 최적화(optimization) 문제의 하나이기 때문이다. 예를 들어 거대한 유조선(oil tank ship)을 건조하는 상황을 가정해보자. 자동차와 달리 유조선을 구성하는 부품은 수십만 개 이상이며 이들은 어떤 규칙에 의해서 조립되어야 한다. 아무것도 설치되지 않은 배, 갑판도 없는 배에 조타실의 부품을 설치할 수는 없기 때문이다. 이 복단다단한 과정의 일정을 최소화하고, 공급 부품이 대기하는 시간[3]도 줄여야 한다. 또한 부품의 조립 순서도 지켜야 하는 등 제약 조건이 매우 많으며 까다롭다.

18.1 Job Scheduling (다양한 조건에 따라서 매우 많은 종류의 문제가 있다.)

이 문제를 구성하는 요소를 보자 JS = <P, C, T, S, O>

- P, 가용가능한 processor의 수
- C, 고객의 수(no. of customers)와 요구 조건 customer
- T, processing time, allocation strategy (배당전략)
- O, Objective function and constraints

18.2 **[실전 문제]** 현재 수강하는 과목은 모두 7개인데 각각 과목에서 하나씩의 과제물이 나왔다. 과제물마다 난이도가 다르다고 가정한다. 즉 각 과제물마다 완성하는데 걸리는 시간은 같지 않다. 어떤 과제는 3일, 어떤 과제는 쉬워 반나절 0.5 days면 충분하다. 그리고 각 과제마다 마감시간도 다르다. 또한 과목마다 과제물의 성적에 차지하는 비율도 다르다. 그리고 과제물마다 각 교수님이 주는 점수의 평균과 분산도 다르다. 어떤 분은 거의 90점 수준에서 부여하고 어떤 분은 0점부터 100점까지 큰 차이를 낸다. 즉 대충해도 대략의 점수를 받을 수 있는 과제도 있고 그렇지 않은 과제도 있다. 자 여러분이라면 어떤 과제물을 선택하여 어떤 순서로 해야 할 것인가? (시작은 1일이며, 남은 시간은 일주일 6일이 그 끝이다.)

과제물	예상시간	반영비율	마감시간	평균점수	우선순위
1	3	10%	2	80	
2	2	30%	3	50	
3	1.5	25%	4	60	
4	0.5	5%	4	40	
5	1	10%	2	70	
6	2	50%	5	90	
7	4	30%	3	50	

3) 필요한 부품이 실제 설치되는 시간보다 너무 일찍 도착하면 조선소 어디엔가에 적치해두어 한다. 만일 이런 부품이 아주 크고 갯수도 많다는 이들을 적치해두는 것 자체가 큰 문제가 된다. 적절한 공간, 관리 인원등이 모두 비용으로 처리되기 때문에 부품을 필요한 시점에 바로 들어와야 한다.

여러분이 가장 시급하게 해야 하는 일을 적절한 목적함수를 결정하는 것이다. 단 하나의 목적함수만을 구성할 필요는 없으며 두 개 이상의 목적함수를 결합하여 이 둘의 평균으로 계산할 수 있다. 예를 들면 $G(x)=1/2 \cdot f(x)+1/2 \cdot t(x)$는 두 목적함수[4]의 값을 반반 섞을 수 있다.

18.3 Multiple-Server Scheduling Problem

가장 단순한 Job Scheduling 문제는 단일 처리기(processor)를 가정하는 것인데 이 문제가 복잡해지면 처리기의 수가 하나 이상, 더 복잡해지면 처리기의 종류가 다른 경우다. 이 경우 목적함수는 다음과 같다.

- **시스템 내부 총 대기시간 최소화하기**
- **얻을 수 있는 이득의 최대 값**
- **최대이득 – 총 비용**

앞서의 문제를 2명(2개의 processor)이 있는 경우를 가정해서 풀어보자. 즉 친구 한 명이 과제를 몰래 도와준다고 가정해보자. 이제는 하루 2개의 과제를 동시에 진행할 수 있다.

과제물	예상시간	반영비율	마감시간	평균점수	우선순위
1	3	10%	2	80	
2	2	30%	3	50	
3	1.5	25%	4	60	
4	0.5	5%	4	40	
5	1	10%	2	70	
6	2	50%	5	90	
7	4	30%	3	50	

좀 더 복잡하게 문제는 만든다면 그 도와주는 친구는 여러분 능력의 2배인 경우이다. 즉 여러분이 이틀 걸릴 일을 이 친구는 그 절반인 하루 처리할 수 있는 경우이다. 이런 상황에서 어떻게 일정을 정리하는 것이 최적일지 생각해보자.

18.4 job마다 다른 이득(예상 학점이 있는 경우)과 마감이 있는 경우의 스케줄 짜기

과목	A	B	C	D	E	F	G
마감일	3	3	2	1	2	1	2
점수	80	65	85	65	82	75	68

욕심쟁이 전략은 매우 간단하다. 목적함수가 이득의 합의 최대이기 때문에 우리는 이득에 “눈이 어두워야” 한다. 따라서 가장 이득(점수)가 높은 과목을 선택해

4) 당연히 둘 모두 max가 되어야 하겠죠 ? 만일 한 개가 min이라면 음수 부호를 붙혀서 max로 바꾼다.

서 처리하되 그 시작 일자는 최대한 미룰 수 있을 때까지 미룬다. 위의 에라면 가장 점수가 높은 C선택하고 이를 최종 마감일인 2일차에 처리한다.(넣는다.)

일자	1	2	3	4
선택		C		
점수		85		

그 다음으로 점수가 높은 E를 선택하는 이 마감일은 2일이고 이미 2일 칸에는 다른 작업이 있으므로 우리는 양보하여 이를 그 앞 1일에 넣는다.

일자	1	2	3	4
선택	E	C		
점수	82	85		

18.5 (on-line 알고리즘의 예)

잘 알려진 TETRIS 게임에서 쌓는 블록을 Greedy하게 찾아 넣기

- 뒤에 들어오는 데이터를 전혀 모를 경우
- 사람 사귀기, 취업하기

18.6 (on-line 알고리즘의 예)

여러분이 어떤 벤처업체의 사장이라고 가정을 해보자. Job Exhibition (취업박람회)가 열렸는데 여기를 통하여 연구원을 뽑기 위하여 준비하고 있다. 오늘은 100명의 연구원이 대기하고 있다. 지원자들에 대한 사전 정보는 없으면 면접하는 시점에 지원자는 자기 소개서와 경력사항 증명서를 제출한다. 여러분은 이 내용을 보고 바로 합격을 시킬 것인지 아니면 거절(reject)할 것인지를 결정해야 한다. 단 한번 reject 한 지원자는 다시 되불러 올 수는 없다.

이 상황에서 100명의 지원자 중에서 최고의 인재를 뽑으려고 한다면 가장 가능성있는 알고리즘은 어떤 것이 되어야 할지 생각해보자. (지원자의 자기소개서에서 어떤 정수가 적힌 자료라고 생각하면 된다. 즉 미래에 어떤 값들이 준비된 지 모르는 상황에서 최고의 값을 선택하는 것이다.) 뽑을 사람은 1명이며 그 사람을 선발하면 그 뒤에 줄을 선 사람들은 모두 다른 곳으로 간다.[5]

결혼시장과 매우 유사하다. 한 사람이 복수의 이성을 만나지 않는 상황을 가정한다. 현재 만나고 있는 사람보다 더 좋은 사람을 만날 가능성이 많다면 지

5) 결혼시장과 매우 유사하다. 한 사람이 복수의 이성을 만나지 않는 상황을 가정해서이다. 만나고 있는 사람보다 더 좋은

금 교제하고 있는 사람을 버리고 다른 사람을 찾아야 한다. 만일 지금 사귀고 있는 사람이 현저하게 수준이 낮은 사람이라고 판단되면 당연히 “후보교체”가 발생한다. 그러나 더 나은 사람, 더 더 나은 사람을 찾아 끊임없이 바꾼다면 궁극에는 이전에 만났던 사람이 결국 제일 좋은 사람이었다는 사실을 알게되고 후회하게 된다. 자 이제 여러분의 알고리즘은 어떤 것일지 한번 말해보자.[6)]

6) https://www.cantorsparadise.com/math-based-decision-making-the-secretary-problem-a30e301d8489
https://johngrib.github.io/wiki/secretary-problem/

Lecture 19 : Huffman Encoding (인코딩)

19.1 정보이론적 관점으로 보는 압축률

- 비손실 압축 (looseless compression)
- 손실 압축 (loosy compression)

  e.g., 사진, 음악(intrinsically impossible)

Q) 여러분의 hwp 파일을 압축하고 또 압축하면 1 byte로 만들 수 있나 ?

19.2 Entropy란 무엇인가 ?

- 다양한 엔트로피의 정의
  - 양자역학적, 정보이론적, 물리적,
- 무질서도 = Entropy(p_i) = $p_i \cdot \log\frac{1}{p_i}$
- Randomness와 Kolmogorov Complexity

Q) 우주에서 오는 신호가 외계인이 보낸 것인지 아닌지를 어떻게 알아낼 수 있을까 ?

19.3 Sound Encoding for MP3

a. 아날로그 신호를 sampling 한다.

b. 양자화(quantization) 작업을 한다. -
  - 인지과정을 고려하여 매우 신중하게 해야 한다.
  - 사진(digital photo)의 경우도 마찬가지

c. 어떻게 하면 이렇게 긴 부호의 sequence를 짧게(경제적으로) 표현할 것인가 ?

19.4 여러분은 다른 별에 있는 자신의 이성 친구에게 문자 메시지를 보내고자 한다. 여러분이 자주 보내는 문자의 종류는 8가지이다. 그런데 이 우주 문자대화 통신비가 상상을 할 수 없을 정도로 비싸 전송 1 bit 당 무려 1000원씩이나 받는다. 따라서 문자를 그대로 치지 않고 뭔가 encoding을 해야 한다.

문자	M1	M2	M3	M4	M5	M6	M7	M8
내용	지금 우주선 타고 집으로 가고 있습니다.	너, 뒤진다 정말!	오늘 실습실에서 프로그램 숙제가 있어서 약속에 못갑니다. 내일 연락 줘	쫌!!!		고마해라!		
비율	0.1	0.1	0.4	0.1				

이 문제를 Greedy하게 해결해보자. 일단 가장 자주 보내는 메시지를 가능하면 짧게 encoding을 하는 것이 유리하다. 단 variable encoding을 했을 경우에 간 word를 구별할 수 있어야 한다. 한 메시지의 encoding이 다른 메시지 문자의 prefix가 되어서는 안 된다.

Ma : 01 Mb: 0110 Mc:0 Md:1101 ...,

19.5 Message Coding with Binary Tree

- Prefix-free Coding : 어떤 코드는 절대 다른 코드의 접두어가 아니다,
  e.g.) 아닌 예: A=1101, B=11, C=01010, D=0
- Prefix-free Coding이 아니면 뭔가 stop word가 필요하다.
- 따라서 각 메시지를 Binary Tree(0,1 bit의 경우)의 leaf에 매달면
  해당 메시지는 그 tree의 path coding이 된다.
- Coding Tree의 성능분석

Q) 가장 긴 코드 (worst case)

Q) 가장 짧은 코드 (best case)

Q) 평균적인 경우 (average case)

$$\text{Cost of Tree} = \sum_{i}^{n,\text{메시지의갯수}} f_i \cdot (depth \quad of \quad symbol \quad s_{i)}$$

19.6 Huffman Encoding 알고리즘의 주요한 착안점(아이디어)

- 자주 안 쓰이는 메시지는 가능하면 coding Tree의 바닥에 배치
- 어떤 메시지 A의 비율이 다른 메시지의 비율보다 낮으면
  coding tree에서 반드시 그 높이가 B보다 높지 않아야 한다.
- 일단 빈도가 낮은 것끼리 먼저 묶고 본다. 이게 당장 가장 유리하기 때문이다. 왜냐하면 먼저 묶는 쌍이 긴 code를 부여받기 때문이다.

19.7 Huffman Coding의 예. 다음 코드를 위한 Huffman code를 제시하고 그 과정을 보이시오.

M	A	B	C	D	E	F	G	H	I	J
f_i	10	7	21	5	8	11	6	4	9	2

M	A	B	C	D	E	F	G	H	I	J	K	L	M
f_i	12	5	13	9	10	7	6	14	9	12	6	4	10

19.8 Huffman Coding의 구체적인 알고리즘과 필요한 자료구조

i) 모든 메시지를 f_i 를 min Priority Queue(PQ)에 모두 넣는다.

ii) PQ에서 Top에 있는 두 개의 원소 E_i 와 E_j를 꺼낸다.

iii) 꺼낸 두개의 노드를 $E_{i,j}$로 합치고, 그 weight를 $f_{ij} = f_i + f_j$ 로 새로 고쳐서 우선순위 큐 PQ에 다시 넣는다.

iv) 이 작업을 Queue가 빌 때까지 계속 진행한다.

19.9 Huffman Coding의 Time Complexity 분석

a. 사용 빈도수에 따른 1차 sorting

b. 나머지는 n번의 Priority Queue processing이므로 Huffman Coding resorts to the complexity of PQ implementation

Time = O(|V|*(a time for priority insert)) = O(2*|V|log |V|)

19.10 Ternary Huffman Coding과 Binary Huffman Coding의 차이점

- N개의 원소가 있을 때 전체 코드의 크기
- 평균 코드의 크기
- 각 구성할 때의 시간 복잡도

19.11 일반적인 Huffman Coding을 프로그램으로 구현할 때의 문제점

- 반드시 전체를 미리 읽어 frequency table을 만들어야 한다.
  (이게 보통 일은 아니다. 예를 들어 수백만개의 심볼이 준비되었다면)

- 만일 real-time으로 기호들이 들어온다면 그것은 불가능하다.
- Dynamic Huffman Coding
  (보통의 압축 프로그램이 쓰는 방법)

Lecture 20 : 배낭문제(bin packing)를 위한 다양한 알고리즘

20.1 배낭 채우기 문제

- 주어진 배낭의 용량을 넘지 않을 때까지 가장 많은 물건을 채운다.
- 우체국에서 여러 가지 물품 선택해서 택배 보내기,
- 비싼 뷔페에서 음식 맛있게 먹기
- S = { 4, 5, 13, 23, 45, 56, 31, 9 }
  S에서 몇 개를 선택하여 K를 넘지 않도록 하되 가장 그 값이 크도록

20.2 배낭 채우기 문제의 형식

- 각 물건마다 가격이 있을 경우, 따라서 물건은 (용량, 가격)의 쌍으로 표시
- 배낭의 용량이 제한된 경우. Capacity
- 목적함수는 배낭에 채워진 물건의 총 가격 혹은 성능

20.3 배낭 채우기 문제의 두 가지 형식

- 0/1 타입(넣든지 안 넣든지, 단 통째로 넣어야 하며 자르면 안됨.)
  - "도둑" 집안을 털어서 용량이 정해진 배낭에 담아 훔쳐 가기

- Partial KnapSack (잘라서 넣어도 가능함)
  - 정육점에서 임의의 무게로 잘라 팔 수 있는 삼겹살의 경우
  - 쌀집(?)에서 무게로 달아서 파는 곡식(Grain) 구입하기

20.4 욕심쟁이 탐색 방법으로 0/1 배낭 채우기 (물건을 다 넣든지, 안 넣든지)

- 원칙) 가장 값이 나가면서 양이 작은 것부터 먹어 치운다.
  - <회> 와 김밥, 빵 중에 무엇부터 먹을 것인가 ?
- Greedy Packing으로 optimal이 되는 배낭 채우기 문제의 예

20.5 Dynamic Programming 접근법으로 배낭 채우기

- Pseudo-polynomial time algorithm (의사 다항시간 알고리즘)
  데이터의 "갯수"가 아니라 그 각각의 "값(value)"에 따라서 복잡도가 결정)

	5	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**.**	**.**	**.**	**57**	**58**	**59**	**60**	**61**
5																
7																
9																
11																
21																
29																

20.6 **[실전문제]**

우리는 어떤 화물을 아주 먼 곳까지 수출하려고 한다. 한 컨테이너를 사용하는 비용은 개당 200만원이다. 그 안을 꽉 채우든 아니든 그건 비용에 상관이 없다. 각 컨테이너의 크기는 2차원의 크기는 10*5 이다. 준비된 물건들의 크기는 3*3, 4*5, 5*5, 6*3, 7*2, 5*5, 5*1, 7*8, 5*4, 3*2, 7*5. 8*2 이다. 이들을 배치할 때 90도, 270도 회전(rotation)도 허용된다. 이 경우 몇 개의 컨테이너를 대여(renting)하면 되는지 그 최소 수를 계산해보시오.