알고리즘 분석 기말고사 교수님 요점정리

1. 우리, 저번에 시험을 봤던 데가 다이나믹 프로그래밍 끝까지였던가요?TSP는 그때는 안했던것 같아요. 혹시 중간고사때 TSP가 들어갔는지 안들어갔는지 아시는분. 안들어갔죠..? 저는 사실. 이것도 일단 익혀놓으면 나쁘지는 않아요. 나쁘지는 않은데, 이거를 내가 저번에 설명할때도 얘기 했지만, 첫번째로, 시험지에 이걸 풀기에는 너무 이게 (???)하는게 양이 너무 많아요.첫번째로, 그리고, 이게 그나마 알고리즘으로서 효율이 좋은거면, 여러분들에게 좀 익혀놓으면 나중에라도 도움이 될테니까 참고사항으로라도 해보라고 할텐데, 이게 사실상 별로 좋은게 못돼서, 실제로 이런문제를 여러분들이 풀어야 할때는 예전에 GA(Genetic Algorithm)이라고, 이전에 앞에서 보여드렸던 그런식의, 계산적인 알고리즘을 동원해야하는것이 일반적이기 때문에, 이것에 대해서는 언급을 따로 안할게요,
2. 탐욕적인 접근방법입니다. 네. 그래서, (??파울??)은 사실, 한쌍이라고 했죠,그래서, 탐욕적인 접근방법에서는, 일단은 제일 먼저 해야하는게 뭐라고 했어요? 우리 동적계획법이나 뭐 이런것들도 각자 제한사항이 있었죠.알고리즘을 적용하기 위해서. 그러면 탐욕적인 알고리즘에서는 제한사항이 뭐였어요? 이걸 적용했을때, 이것이 과연 알고리즘을 통해 해답을 내는지. 그것에대해 먼저 검증을 해야 한다고 했죠. 이걸 어느 단위로 해야 한다고? 인스턴스 단위로 해야한다고 했어요. 문제 사례에 대해서, 예를 들어 "거스름돈 문제는 탐욕적 알고리즘으로 풀수있어!"라고 한다면 그 말은, "틀렸다!"라고 답할 수 있어야 해요. 어떤 거스름돈 문제에서는 풀 수가 있는데, 거스름돈이 막 이상한 화폐 단위를 사용하는 국가에 탐욕적인 알고리즘을 적용해서 거스름돈 문제를 풀겠다! 하면 그것은 "안된다."라고 얘기를 했었죠.
3. 그래서 탐욕적인 알고리즘을 통해서 푸는것이, "Prim 알고리즘" 그쵸? 와, 이렇게 보니까 오랜만에 보는 것 같은. 그런 기분이 들어요~,프림스 알고리즘,여러분들 한번씩 다 구현해 보셨을 테니까 다들 알겠죠? 여러분들 증명을 열심히 제가 얘기해줬는데, 설명할때도 얘기했지만, 이걸 시험문제로 내기에는 양이 너무 많아요.길기 때문에, 여러분들 마음속에만 담아 두는걸로.
4. Kruskal Algorithm. 크루스칼 알고리즘은 중요한게 뭐였어요? 서로소 집합에 대한 내용이었죠?그래서 서로소 집합에 대한 내용은 특히, 뒷부분에서 따로 한번 다뤄 주고있었죠.이거 자체로도 하나가 또다른 문제의 유형이라고 했어요. 서로소 집합을 어떤 식으로 관리를 해 줄 것이냐는것. 그래서, 이런 부분에 해서, 여기서 나오는 Ackerman function 뭐 이런 것 까지는 알 필요가 없고. 그냥 이것들이 어떤식으로 동작을 한다. 어떻게 이것이 Cycle 인지 아닌지. 발생하는 것을 관리를 한다라는 부분. 이런것을 조금 알아 둬야 하겠어요.서로소 집합에 대한 것들.
5. 단일출발점최단경로문제. Dijkstra 알고리즘 이었죠.네. 여러분들이 과제로 풀었기 때문에 이거는 시험문제로 내면 여러분들이 다 잘 풀것이라고 믿어 의심치를 않아요. 그냥 그렇게 넘어갈게요.
6. HuffmanCode. 허프만 코드! 이거, 제가 수업할때 얘기했던것 같아요. 이거, 단순한데, 문제내기 참 좋다~ 오해할 여지도 없고, 문제내기 참 좋은, 그런 내용이다. 라고 얘기를 했었어요. 그리고 이거 어디에 쓰인다고 했어요? 단순히 이렇게 제가 얘기 하니까 달달달 외우는 경우들이 있던데, 이거 압축같은거 할 때 쓰인다고 했죠. 이거 얘기할때 여러분들이 별로 달가워 하지 않았었지만, (???압축과는 다른 성격인???) Arithmetic 압축기법이나 뭐 이런 걸 조금 했었죠. 허프만 코딩의 단점은 1bit를 통해서 최소 하나의 정보밖에 표현을 못한다. 그런내용.
7. 0-1knapsack problem. 우리가 방금 전에 풀었던 건데 여기서도 한번 처음으로 등장을 했죠. 탐욕적 알고리즘을 통해서는 이걸 못푼다고 했어요. 대신 뭘 풀 수 있다구요? Fractional knapsack problem은 풀 수 있다고 했죠? 그쵸? 근데, 이거는 앞에서도 설명을 했지만, 우리가 결국 Branch and Bound에서 특히나 0-1 배낭문제를 풀고 있기 때문에 거기의 설명에 포함이 되어있기 때문에 이걸 따로 뭔가 따로 꺼내가지고 얘기하기가 조금 어려워요.
8. 비교되어있는 개념같은 건 그냥 알아두시면 될 것같고. 네. 이게 또다른 문제라고 했죠? 상호배타적 조합. 집합에 대해서 어떻게 관리할 것인지. 방법이 여러개가 나왔는데, 여러분들이 물론, 각각에 대해서 모두 이해하고 있다면 좀 좋겠지만. 보통 알고리즘 책에서 같은 문제에 대해서 여러가지 방법을 제공하면 제시한 방법중에 가

장 나중에 나온 방법이 보통 제일 좋았다. 라고 얘기했죠. 마지막 방법인 tree(forest) 방법이 왜 좋은지. 어떤 것들을 활용 했기에 더 좋은건지.방법을, 사용하는 방법을 실제로 잘 알아두면 좋겠죠. 두가지 방법을 사용했죠. weighting-union 방법과 path compression 그죠?

9. 범위가 생각보다 적어서, 이거...
10. Backtracking 되추적, 깊이우선 검색. n-queens Problem. 네. 4-Queens Problem 이나 n-Queens Problem 이나. n-Queens Problem이 일반화돼서 나온거니까, 굳이 4-Queens Problem이 나올 필요는 없겠죠. 굳이 나온다면 n-Queens Problem이겠지. 근데, 이거는 그냥 사실 상태공간트리에 여러분이 조금 익숙해 지라는 그런 느낌으로 해서, 좀, 여러분들이 조금 그나마 낯설지 않은 문제로 해서 예제로 나왔던 것이고, 사실 이것을 아주 그냥 달달달 외울 필요까지는 없을 것 같아요. 그나마 꼭 익혀보라고 한다면, 이런, 부분집합의 합!
11. 부분집합의 합! 이런거 같은 것은 익혀둘만 한데. 이거는 그... 0-1 배낭문제를 푸는 것과 많이 유사해요. 그래서 0-1 배낭문제를 제가 출제를 한다면 이 부분집합의 합에 대해서는 출제를 안하거나. 하면서 두가지가 모두 출제하기는 조금 어려운 그런 관계에 있다. 그게 왜 중복되는 속성을 가지는 것에 대해서는 제가 몇가지 설명을 했었죠?그죠? 예를 들자면, 저걸 아이템들을 다 정렬을 해 놓고 시작을 하죠? 저걸 왜 저렇게 정렬을 해 놓고 시작을 했어요? 그래야 유망성을 빨리 판단해서 잘라버릴 수가 있기 때문이요? 아이템들을 정렬을 안해놓고 시작하면, 바닥까지 다 봐야하죠. 그죠? 마치 우리가 정렬 안해놓은 데이터 집합에 대해서는 우리가 순차검색이 검색중에 효율이 별로 좋지 않다는 것을 알고 있어도 순차검색밖에 못쓰는 것처럼. 정렬이 되어있으면 하다 못해 이분검색이나 그런 것들을 쓸 수 있죠. 그런데 그렇게 못한다는 거죠.
12. 그래프 색칠하기 ! 여러 (???)에 대한 얘기들이 많이 나왔었어요. 그래프는 요지경 세상이야~ 뭐 그런 얘기까지 나왔는데, 네. 그것에 대한 것은 그냥 적당히.
13. Hamiltonial circuits. 해밀토니안 회로. 구성하고 푸는것. 경로에서 이미 방문한 것인지 확인하는 것. 제가 여기서 뭐 그렇게 중점적으로 얘기 했던 것은 아니기 때문에.
14. 정 문제 낼것이 없다면, 중간고사 범위에서 한 문제 낼까요? 여러분들이 숙제한 것중에서 한문제 내는거 어때요. 어챠피 숙제한 범위가 중간고사에도 걸쳐 있잖아요. 그건 좀 아닌가요? 알겠어요 ㅎ
15. 자. 아무튼, 되추적 알고리즘이라는 것을 드디어 넘어 가서, Branch and Bound로 오고 있죠. 그죠?
16. 0-1 배낭 채우기 문제. 우리가 여기서 0-1 배낭채우기 문제를 거의 절반 이상 풀었어요. 이 강의자료가 거의 다 이걸로 이루어져 있죠. 너비우선, 최고우선, 깊이우선, 뭐 이런식으로, 그래서, 우리가 사실 이런 것들에 대해서 여러분들이 전공이 아니고 비전공. 뭐 그런식으로 배우다 보니까 자료구조임에도 불구하고 자료구조 원리에 대한 내용은 많이 못배우고 프로그래밍 실습같이 객체지향 절반, 프로그래밍 기초 절만, 이런느낌으로 배운다는 것을 조금 들어서. 사실 자료구조라는게 사용하는 것에 따라서 어떤 효과가 나오는지. 이런 것 좀 느낄 수 있는 그런 예가 아니었을까 싶은데, 근데, 이번 것에 대해서 막 모든 방법을 다 알아봐라! 하는 것은 조금 아닌 것같아요. 사실 여러분들이 모든 방법들을 다 알아보지 않는다고 하더라도, 그 중 한가지만 알고 있으면, 자료구조에 넣고 빼는 거니까 다들 어떻게든 돌려볼 수가 있잖아요. 그죠?그러니까 그건 별로 의미가 없다고 생각을 하고. 굳이 하나만 알아본다고 하면 뭐가 있겠어요? 그나마 가장 좋은게 최고 우선이잖아. 그죠? 그러면 최고우선검색 정도는 알아 두어야 겠죠?물론 여러분들이 이걸, 제대로 구현한다고 하면, 매번, 최고인 아이템을 꺼내기 위해서 꺼낸다음에 다시 정렬하고정렬하고 그러는 것은 굉장히 불합리하기 때문에, 이것은 실제로는 힙(heap)을 이용해서 사용을 한다. 라는 것인데, 그러면 "힙"을 어떻게 구현해요? 라고 물어볼 수 있는데, STL이라는 게 있다고 했어요. STL에 이미 Heap 이나 Vector나 기타 등등 뭔가 다양한 것들. 그런 것들을 이미 제공을 하고 있기 때문에. 그걸 이용해서 한번 구현해 보면 된다.Priority Queue도 마찬가지에요. 거기서 다 구현된 것을 제공해 주고 있다.
17. 분기한정법. 최고우선검색 알고리즘. 경로가 아직 유망한지 아닌지를 점검을 해서 뭐 그렇게 한다. 이것정도는 알아두면 좋다라고 했죠.제가 원래 옛날같으면은 이런문제를 제가 기말고사때 거의 안냈는데, 이번에는 범위가 너무 좁다 보니까, 어쩌면 코드의 빈칸채우기. 중간고사때 나왔던 그런 문제가 한 문제 정도는 나올 수 있을 것 같아요.
18. 그래서, 이런 분기한정 가지치기로 0-1 배낭문제를 푼다 라는것은, 이런식으로 트리를 구성하고 방문순서 같은 것을 저희가 작성을 하고. 이 (??집엔지로??)되는 과정, 제가 앞에서 이렇게 그렸던 것 처럼, 이런식으로 표현할 수 있을 정도만 되면 돼요.

19. 외판원 문제를 우리가 풀어 봤어요. 근데, 앞에서 설명했다시피, 알고리즘의 효율이 그다지 좋지는 않아요. 그래서 우리는 그냥 Branch and Bound를 이용해서 다른 예제들을 하면서 그냥 묻혔다(?) 의 느낌으로 그냥 끝내면 될 것 같아요.
20. 그래서 여기까지 해서 끝내면 사실 어때요? 여러분들. 범위가 상당히 좁죠? 그래서 걱정이긴 한데, 그렇다고 여기서 갑자기 7장을 나갈수는 없잖아요. 그쵸? 일단 여기서 끝내고. 정 안되면 시험 하나하나당 배점이 올라가는 것밖에 없겠죠. 그렇게 해서 수업이 오늘까지로 마무리가 될 것같은데.
21. 끝.