22100062 권혁주

a. 과제 수행 중 겪은 어려움: 프로그램이 큰 데이터 사이즈 특히 제 코드에선 100부터는 시간이 너무 오래 걸리는 문 제가 있었습니다. 특히 Brute Force계산 시간이 급격히 늘어나기 때문에 더 효율적인 알 고리즘이 필요함을 깨달았습니다. 다양한 알고리즘을 구현하는 과정에서 각 알고리즘의 특성을 이해하고 최적화해야 했습 니다. 특히 메모리나 시간 제약을 고려한 알고리즘 선택이 어려웠습니다. 그리고 블로그 나 자료에서 본 코드를 녹여내는 것이 너무 어려웠습니다. 아이템의 무게나 이득이 0인 경우 등 예외적인 입력을 처리하는 데 어려움이 있었습니다.

b. 원래 코드 아이디어와 참고한 코드의 차이점:

원래 코드 아이디어:

그리디 알고리즘을 사용하여 이득/무게 비율을 기준으로 아이템을 선택하려 했습니다. 그러나 이 방식은 **fractional knapsack**에만 적합하고, **0-1 knapsack**에는 부적합한 점이 있었습니다. **동적 계획법(DP)** 방식은 2차원 배열을 사용해야 하므로, 메모리 사용이 많고 코드가 복잡했습니다. **분기 한정(Branch and Bound)** 방식은 초기 구현에서 제외했으나, 나중에 최적화를 위해 추가하려는 계획이었습니다.

참고한 코드

동적 계획법(DP)을 우선적으로 사용하여 큰 데이터에서도 효율적으로 문제를 해결하는 방식을 채택하였습니다. 이 방식은 특히 큰 데이터셋에서 효율적이며, **메모리 사용과 시간 복잡도**를 최적화할 수 있습니다.

그리디 알고리즘은 이득/무게 비율을 기준으로 아이템을 정렬하여 선택하는 방식으로 구현되었습니다. 이 방식은 **fractional knapsack** 문제에 적합하며, 아이템의 이득과 무게를 정확히 고려한 선택을 할 수 있습니다

Branch and Bound 방식을 활용하여 탐색을 효율적으로 제한하고 우선순위 큐를 사용하여 최적의 해를 빠르게 찾아낼 수 있었습니다. 이를 통해 브루트 포스 방식보다 성능을 크게 개선할 수 있었습니다.

온라인 코드에서는 chrono를 사용하여 각 알고리즘의 실행 시간을 측정하고 성능을 비교하는 코드가 포함되어 있었습니다. 이를 통해 각 알고리즘의 효율성을 객관적으로 평가할 수 있었습니다.

c. 이번 과제를 통해 배운 것: 작은 데이터에는 브루트 포스가 유효하지만, 큰 데이터에는 동적 계획법이나 분기 한정 과 같은 효율적인 알고리즘을 사용해야 한다는 것을 배웠습니다. 알고리즘의 성능을 측정하고, 입력 크기에 따라 성능이 어떻게 달라지는지 평가하는 방 법을 배웠습니다. 데이터의 특수한 경우(예: 0인 값 처리)에 대한 처리가 중요하다는 점을 깨달았습니다. chrono를 사용해 알고리즘 실행 시간을 측정하고, 성능을 최적화하는 방법 을 배웠습니다. 이번 과제를 통해 알고리즘 설계와 성능 최적화에 대한 중요한 경험을 얻었습니다.