**Computer Algorithm**

**Chapter 10**

**Greedy**

**[실습] Greedy Algorithm구현**

**실습 목표**

* 그리디(탐욕) 알고리즘을 이해한다.
* 그리디 알고리즘의 이해를 바탕으로 문제를 해결하는 과정에서 필요한 알고리즘을 구현할 수 있도록 한다.

**요구사항**

* **실습과제 1) (100점) 그리디 알고리즘을 구현한다.**

**1-1) (40점)**

주어진 용량의 물통을 300ml와 500ml의 컵을 사용하여 물을 채우려고 한다. 이 때 최소한의 횟수로 컵을 사용하여 물을 다 채울 수 있는 횟수를 구하는 프로그램을 구현하여라. 만약 주어진 컵으로 용량의 물통을 정확히 채울 수 없다면 -1을 리턴한다.

컵은 300ml와 500ml의 두 사이즈 만을 사용하며 채워야 하는 물통의 크기는 입력 파일로 주어진다.

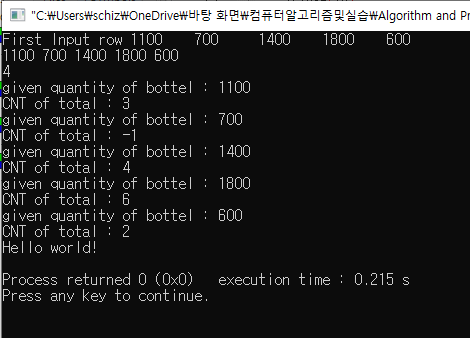
입력은 파일에서 받는 것으로 하며 파일이름은 “input.txt”로 한다. 입력파일의 형식은 아래와 같으며 숫자들은 공백으로 구분된다. 출력은 아래와 같이 하도록 하며 컵을 사용하는 횟수를 나타내도록 한다. 만약 작은 컵을 사용하여 물통의 용량을 정확히 채울 수없다면 -1을 출력한다.

입력 파일 형식: 1100 700 1400 1800 600

출력 형식: 3 -1 4 4 2

위 입력은 300ml, 500ml 두 작은 컵으로 1000 ml, 700ml, 1400ml, 1800ml, 600ml의 물통에 물을 다 채우는 데 필요한 작은 컵의 사용횟수를 출력하도록 하는 데 있다.

1100 일 경우 500ml 1컵, 300ml 2컵으로 총합 3컵, 1400은 500ml 1컵, 300ml 3컵 총합 4컵으로 채운다.



**Hint)**

최소한의 횟수로 다 물을 채워야하기 때문에 큰 컵으로 먼저 채우도록 한다. 왜냐면 큰 컵을 선택하면 작은 컵을 선택하는 경우에 비해 더 적은 횟수로 물통을 채울 수 있기 때문이다. 따라서 이 방법에서는 최대한 500ml컵으로 채워보고 남는 곳에 대한 용량을 300ml컵으로 채우는 방식을 생각해볼 수 있다. 아래 코드는 이를 위한 슈도코드의 일부이다. 아래 코드를 참고하여 프로그래밍을 하도록 한다.

|  |
| --- |
| for (i = 물통용량 / 500; i >= 0; i--) {  if (( ) % 300 == 0) {  // cup\_\*00 = \*00ml 컵 사용 횟수  cup\_300 = ( ) / 300  cup\_500 = ( )  }  }  print(cup\_300 + cup\_500) |

**1-2) (60점)**

x축 위에 최대한 많은 직선(선분)을 겹치지 않도록 올려두려고 한다. 입력되는 값은 시작점, 끝점 쌍 으로 입력 받게 된다

입력값은 아래와 같고, 배열에 초기화를 통해 선언한다.

입력파일의 형식은 아래와 같으며 숫자들은 공백으로 구분된다. 또한 출력의 형식은 아래와 같다.

**1 4 2 3 1 2 4 6 3 7 6 8 7 10 7 8 7 11 9 13**

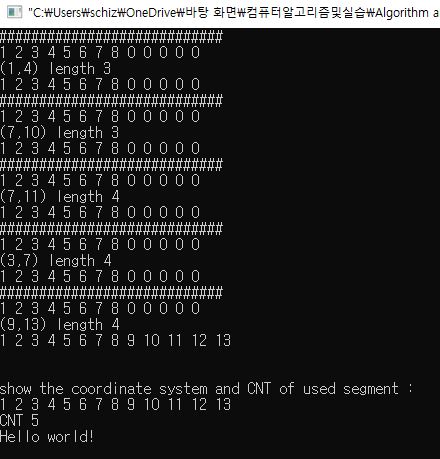
출력결과 : 5

입력파일은 입력이 **1 4 2 3 1 2 4 6** 라고 했을 때 입력되는 선분들은 (1,4), (2,3), (1,2), (4,6)이며 아래 그림과 같이 입력되는 것을 의미한다.

시계이(가) 표시된 사진

자동 생성된 설명

직선들을 겹치지 않게 올리는 것이 목적이므로 위의 그림에서 정답은 (1,2), (2,3), (4,6)의 3개가 된다.

 **Hint)**

가능한 많은 선분들을 올리는 것이 목적이므로 우선은 작은 길이의 선분을 먼저 올리는 방법을 생각할 수 있다. 작은 길이의 선분을 먼저 선택하는 이유는 작은 길이의 선분을 선택하면 확률적으로 더 많은 선분을 남은 직선에 올릴 수 있기 때문이다.

위의 예에서 보면 우선은 가장 작은 길이의 선분인 (1,2) 또는 (2,3) 중에 하나를 선택하도록 한다. 이를 남은 선분들에 대해서도 반복적으로 실행하는 데, 가장 작은 선분이 기존에 선택된 선분들과 겹치는 부분이 있는지를 확인하고 겹치지 않을 경우에 선택을 하도록 한다.

남은 선분들 중에서 가장 작은 길이의 선분을 선택하기 위해서는 선분들을 미리 정렬한 후 선분들을 작은 순서대로 선택하는 방법을 사용해야 하는데, Heap 구조를 사용할 수 있다.

**배경지식**

그리디 알고리즘 (Greedy Algorithm; 탐욕 알고리즘)은 전체적 또는 장기적인 효율을 생각하지 않고 문제해결의 명령어의 시행 시점에서의 최선의 선택을 하여 문제를 해결하는 기법이다. 각 단계에서의 최선의 선택이 장기적으로 최선의 선택이기를 바라는 의도를 가지는 알고리즘으로, 전체적이거나 또는 장기적으로 최선인 방법을 모르거나 또는 찾기 어려울 경우에 사용하는 방법으로 사용된다.

그리디 알고리즘은 알고리즘을 설계하는 데 있어 현재 상황에서 최선의 선택을 한다는 점에서 단순하게 알고리즘을 설계할 수 있으므로 알고리즘 설계가 쉽다는 장점을 가진다. 또한 그리디 알고리즘은 최적의 방법을 찾는 경우도 있다. 하지만 장기적으로 낮은 효율을 보이는 경우도 있으며 그리디 알고리즘은 대체로 단순하게 구성되며 따라서 사용될 수 있는 범위가 제한적이다.

**제출방법**

* 보고서 작성방법: 실습문제 번호별로 결과가 나온 화면의 내용을

캡쳐하여 보고서에 붙여 놓는다.

* 소스코드의 파일이름에 연습문제 번호를 붙이는 것을 잊지 않는다. 예) ex-1.c, ex-2.c
* 결과 보고서에 이름과 작성 날짜를 기입하는 것을 잊지 않는다. 예) 김웅섭\_2020\_09\_01.doc
* 실행결과를 보고서에 작성하여 소스코드와 함께 제출한다.
* 제출 마감 : e-class 제출 마감시간까지