알고리즘 : 동적계획 Dynamic

Idea!
① 작은 문제의 최적값을 저장해 둔다.
② 그 다음 단계의 최적값을 계산.

[관련 교과 : 고등학교 정보]

주제 : 가장 저렴한 가격을 찾아라!!

정보영재교육 20147592
임석영

1. 주제 개요

최적화 문제는 수학 혹은 컴퓨터 과학에 모든 테스트 케이스에 대해 답을 찾는 최적의 해법을 찾는 문제를 말한다. 물건을 구매하는 경우에 같은 물건을 다른 사람보다 비싸게 구입해서 속상했던 경험이 있을 것이다. 단편적으로 한 개의 물건 가격을 쇼핑몰마다 비교해서 구입하는 경우는 몇 번의 클릭과 비교를 통해 그 중 가장 저렴한 물건을 구입할 수 있지만, 여러 개의 물건 또는 상황에 따라 다른 요금을 조합해서 가장 싼 가격 또는 최적의 가격을 찾아서 구입을 하기란 생각보다 쉽지 않다. 방법을 알고 있다 하더라도 최적의 가격을 찾아내는데 많은 시간과 노력이 들어간다. 이러한 복잡한 문제를 해결하기 위한 알고리즘 설계와 프로그래밍을 통해 최적화 문제 해결에 대해 알아보자.

2. 학습 목표

가. 인지적 목표

- 컴퓨팅 사고력을 적용한 문제 해결 절차에 대하여 설명할 수 있다.
- 추상화 과정을 통해 문제를 해결할 수 있는 형태로 표현하고, 프로그래밍을 통하여 자동화할 수 있다.
- 최적화 해법을 찾아내기 위한 문제 해결 기법을 알고 문제에 적용할 수 있다.
- 프로그래밍을 통해 최적화된 해법을 찾아내기 위해 자료형, 변수, 배열, 제어문의 의미를 설명하고 프로그래밍으로 구현할 수 있다.

나. 정의적 목표

- 일상생활 속에서 접하는 문제를 컴퓨팅 사고력을 통하여 효율적으로 해결하는 태도를 가진다.
- 주어진 문제를 해결할 때 다른 사람들과 협업하기 위해 노력한다.
- 문제를 해결하는 방법에는 여러 가지 있음을 알고 주어진 자원을 효율적으로 해결할 수 있는 방법을 찾으려는 노력한다.
- 다른 학생들의 문제 해법을 자신의 해법과 비교하며 문제 해결에 다양한 관점이 있음을 공감한다.

3. 컴퓨팅 사고력 적용 문제

○ 상황제시

올해 새로 개장한 K 스키장은 경쟁 스키장보다 저렴한 리프트 가격으로 많은 스키어와 보더에게 인기가 높다. 리프트 가격을 1일권부터 최대 10일권까지 구분하고 각각의 가격을 차별해 책정했기 때문이다. K 스키장에서 제시한 리프트 가격은 다음의 표와 같다.

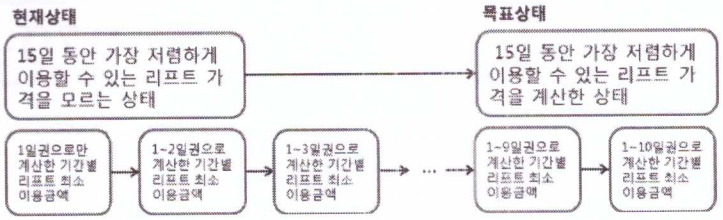
K 스키장 리프트 가격

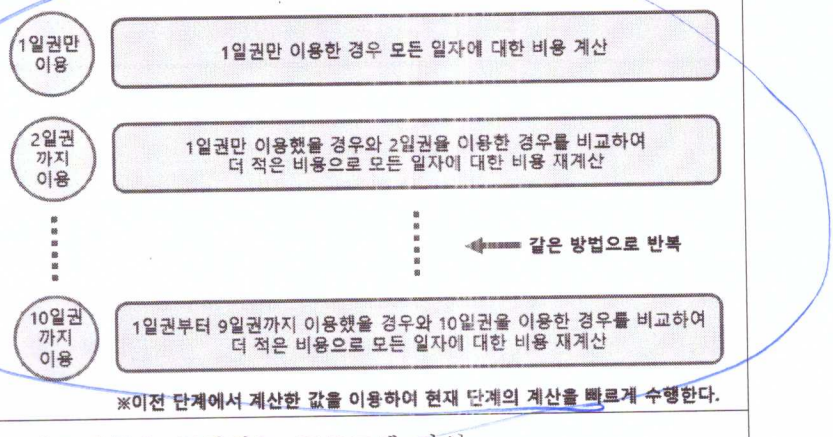
(단위 : 천원)

1일권	2일권	3일권	4일권	5일권	6일권	7일권	8일권	9일권	10일권
12	21	31	40	49	58	69	79	90	101

겨울 방학을 맞이한 정우는 15일 동안 K 스키장에서 친구들과 놀 계획을 세웠다. 그런데 K 스키장의 리프트권 중에는 15일권이 없고, 정우는 최소의 비용을 들이고 싶다. 15일 동안 이용할 수 있는 가장 저렴한 리프트 비용은 얼마일까?

4. 컴퓨팅 사고력 적용 요소

CT 세부 역량	CT 구성 요소	수행 과제
추상화	자료 수집 및 분석	· 현재 상태 – 15일 동안 이용할 리프트권의 최소 비용을 모르는 상태 – 1일권부터 10일권까지의 리프트권 가격을 알고 있는 상태 · 목표 상태 – 15일 동안 이용할 리프트권의 최소 비용을 구한 상태 · 목표 상태에 도달하기 위하여 수행해야 할 작업 – 1일권부터 10일권까지의 리프트권을 적절히 조합하여 15일 동안 이용할 리프트권의 최소 비용 찾기
	핵심 요소 추출	· 스키장 사용 일수 · 각 리프트권의 가격 · 리프트권을 어떻게 조합하느냐에 따라 총 비용이 달라진다.
	문제 분해 및 모델링	·1일권만 이용하는 경우, 1일권과 2일권을 이용하는 경우, 1일권부터 3일권까지 이용하는 경우, …, 1일권부터 10일권까지 이용하는 경우로 문제 분해 

추상화	알고리즘	· 1일권부터 N일권까지만을 이용하였을 때의 일별 최소 비용을 계산하는 행위를 주어진 N에 대하여 반복 수행하며 동적으로 표(배열)를 채워나가다가 최종적으로 원하는 기간동안의 최소 리프트 비용을 구하는 알고리즘을 설계 
자동화	프로그래밍	· 알고리즘을 구현하는 프로그램 작성
	시뮬레이션	· 프로그램을 실행하고 결과를 분석 · 조별 발표 및 공유

5. 차시별 수업 계획

차시	주제	내용	CT요소
1	문제 정의 및 이해	· 문제 이해 및 분석 – 현재 상태 및 목표 상태를 설정한다. · 문제와 관련하여 필요한 자료를 수집하고 분석 한 후 이해하기 쉬운 형태로 표현한다. · 문제 해결을 위해 불필요한 요소를 제거하고 핵심요소만을 추출한다. · 복잡한 문제를 해결하기 쉬운 형태로 변화하기 위해 작은 문제로 분해하고 분해된 문제와 큰 문제간의 관계를 분석한다.	자료 수집 및 분석, 자료 표현, 핵심요소 추출, 문제 분해
2	문제 해결 방법의 설계	· 문제를 효과적으로 해결하기 위한 알고리즘을 설계한다.	알고리즘
3	문제 해결 방법의 구현	· 설계된 알고리즘을 프로그래밍 언어(C언어)를 통해 구현하고 실행결과를 확인한다.	자동화 시뮬레이션
4	발표, 결과 분석 및 평가	· 알고리즘 설계와 프로그램 실행 결과를 중점으로 발표하고 자기평가 및 동료평가를 한다. · 다른 학생들의 발표와 자신이 발표한 내용을 비교하여 알고리즘의 효율성을 비교한다.	시뮬레이션

6. 평가 계획

순서	평가기준	방법
1	문제 상황을 제대로 이해하고 있는가?	관찰평가
2	자료를 제대로 수집하고 분석하고 있는가?	관찰평가
3	주제를 잘 이해하고 핵심 문제를 찾아냈는가?	관찰평가
4	문제를 해결하는 알고리즘을 잘 구성하였는가?	관찰평가, 실기평가
5	프로그래밍을 통해 문제를 해결하였는가?	관찰평가, 실기평가
6	주어진 문제를 해결하는 과정에서 다른 사람들과 협업하려고 노력하였는가?	관찰평가

7. 차시별 지도계획(3차시 프로그래밍을 통한 구현)

교과	정보	영역	문제 해결 방법과 절차
시간	50분	대상	OO고, 1학년
대주제	동적프로그래밍을 통한 최적의 해 구하기		
본시제재	설계된 알고리즘을 프로그래밍으로 구현하기		
학습목표	문제를 해결하기 위한 최적의 해를 구하기 위한 프로그래밍을 통해 구현할 수 있다.		
학습자료	교사용	컴퓨터, 프로젝터, 프리젠테이션 자료	
	학생용	컴퓨터, 필기구	

학습과정	교수-학습 활동	시간	자료 및 유의점
도입	◦ 전시 학습 및 본시 학습목표 제시	5	
전개	◦ 지난 시간에 설계한 알고리즘을 바탕으로 프로그래밍 언어를 통해 구현한다. ◦ 프로그래밍으로 구현하는 과정에서 표현 방법에 대해 질문이 있으면 교사, 주변 학생들에게 질문을 통해 해결하도록 한다. ◦ 시뮬레이션을 통해 설계한 알고리즘과 프로그래밍이 문제를 해결하기 위한 최적의 방법인지 확인한다.	40	설계한 알고리즘을 통해 문제가 해결되지 않으면 알고리즘을 수정하도록 한다.
정리	◦ 결과물 정리 및 제출 ◦ 차시 예고		

8. 문제 해결 방법의 설계 및 결과물(예시)

• 15일 동안 이용할 가장 저렴한 리프트 가격 구하기

① 기간별 가장 저렴한 리프트 가격을 계산하기 위해서는 우선, 기간별 금액을 가장 큰 금액으로 초기화한다. 따라서 1일권만 사용한 경우로 기간별 금액을 초기화한다.

-1일권으로만 15일 동안 리프트를 이용했을 때의 기간별 최소 비용 계산하기 (단위: 천원)

1일	2일	3일	4일	5일	6일	7일	8일	9일	10일	11일	12일	13일	14일	15일
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180

▸ **1일 동안 이용할 가장 저렴한 리프트 비용이 확정된 상태**

② 다음으로 이틀 동안 스키장에서 리프트를 이용하는 방법은 두 가지이다. 2일권을 1장 사거나 1일권을 2장 사는 방법이다. 따라서 2일권 리프트 가격 21,000원과 1일권 2장 가격인 24,000원을 비교하여 작은 값인 21,000원으로 결정한다.

- 1일권과 2일권으로 5일 동안 리프트를 이용했을 때의 기간별 최소 비용 계산하기

1일	2일	3일	4일	5일	6일	7일	8일	9일	10일	11일	12일	13일	14일	15일
12	21	33	42	54	63	75	84	96	105	117	126	138	147	159

▸ **2일 동안 이용할 가장 저렴한 리프트 비용이 확정된 상태**

③ 3일 동안의 경우는 3가지 경우가 있다.

· 1일권 × 3매 = 12,000 × 3 = 36,000

· 2일권 × 1매 + 1일권 × 1매 = 21,000 + 12,000 = 33,000

· 3일권 × 1매 = 31,000

3가지 경우 중 가장 저렴한 31,000원으로 결정한다.

- 1, 2, 3일권으로 15일 동안 리프트를 이용했을 때의 기간별 최소 비용 계산하기

1일	2일	3일	4일	5일	6일	7일	8일	9일	10일	11일	12일	13일	14일	15일
12	21	31	42	52	62	73	83	93	104	114	124	135	145	155

▸ **3일 동안 이용할 가장 저렴한 리프트 가격이 확정된 상태**

④ 위의 과정을 10일권까지 반복한다.

- 1 ~ 10일권으로 15일 동안 리프트를 이용했을 때의 기간별 최소 비용 계산하기

1일	2일	3일	4일	5일	6일	7일	8일	9일	10일	11일	12일	13일	14일	15일
12	21	31	40	49	58	69	79	89	98	107	116	127	137	147

※ 3일 동안의 최소 비용을 구하기 위해 1일권, 2일권, 3일권 중 어느 것을 사용해야 할지 비교하는 경우를 살펴보자. 이미 2일 동안의 최소 비용을 구하는 과정에서 1일권, 2일권을 사용할 때의 각 일차별 최소 비용을 구해서 배열에 저장해 두었으므로 그 값과 3일권을 사용할 때와 어느 것이 더 저렴한지 비교한다.

<알고리즘 표현>

1. 리프트권 종류별 가격을 저장할 [티켓가격]배열을 선언한다.
2. 각 일차까지의 최소 비용을 저장할 [최소비용] 배열에 저장한다.
3. 스키장 이용 일수, 현재 선택한 티켓을 저장할 변수 [기간], [현재티켓]을 선언한다.
4. 1일권부터 10일권까지 각 리프트권의 가격을 입력 받아 [티켓가격] 배열에 저장한다. 스키장 이용 기간을 입력 받아 [기간] 변수에 저장한다.
5. 1일권으로만 리프트권을 구매한 것을 최소비용이라 가정하고 [최소비용] 배열의 값을 초기화한다.(1일권의 가격이 12,000원이라고 하면 2일권의 가격은 24,000원, 3일권의 가격은 36,000원, 10일권의 가격은 120,000원이 된다.) 이는 1일째까지의 최소 비용이 확정된다.

6. 2일째부터 스키장 이용일(i)까지 다음의 ①, ②, ③, ④ 과정을 반복한다.
   ① [현재티켓]에 1일권 가격을 저장한다.
   ② [최소비용]배열의 i번째 값보다 [현재티켓] 값이 작다면 [최소비용] 배열의 i번째 값을 [현재티켓] 값으로 갱신한다. i일째까지의 최소 비용이 확정된다.
   ③ i+1일째부터 스키장 이용일(j)까지 다음의 ㉠, ㉡ 과정을 반복한다.
      ㉠ [최소비용] 배열의 j번째 값보다 [최소비용] 배열의 j-1번째 값과 [현재티켓] 값을 합한 값이 더 작다면 [최소비용] 배열의 j번째 값을 [최소비용] 배열의 j-1번째 값과 [현재티켓] 값을 합한 값으로 갱신한다.
      ㉡ j를 1만큼 증가시킨다.
   ④ i를 1만큼 증가시킨다.
7. [최소비용] 배열의 [기간]번째 값을 출력한다.

<소스코드>

```
#include <stdio.h >
#define TICKETS 10                  // 리프트권의 종류
#define MAX_DAYS 10000              // 스키장 이용 최대 일수
int price[TICKETS +1];              // 리프트권 종류별 "가격" 저장 배열
int minPrice[MAX_DAYS +1];          // 각 일차까지의 "최소 비용" 저장 배열
int main(){
    int days, setPrice;             // 스키장 이용 일수, 선택된 티켓 비용
    int i, j, k;                    // for문의 제어 변수
    for(i =1; i <=TICKETS; i ++)    // 1일권부터 10일권까지 각 리프트권의 가격을
        scanf("%d", &price[i]);     // 입력 받아서 price 배열에 저장
    scanf("%d", &days);             // 스키장 이용 일수 입력 받아서 days에 저장
    for(i =1; i <=days; i ++)       // 1일차부터 구하고자 하는 일차까지
        minPrice[i]=i *price[1];    // 1일권 가격*해당일수로 "최소 비용" 배열 초기화
    printf("기간별 리프트 최소 이용 금액(단위: 천원)₩n₩n");
    for(i =1; i <=days; i ++)       // 필드명 출력
        printf("%2d일 ", i);
    printf("₩n=================================================₩n");
    for(i =1; i <=days; i ++)       // 1행에는
        printf("%3d  ", minPrice[i]);  // "최소 비용" 배열의 초기 상태 출력
    printf("₩n₩n");
    for(i =2; i <=TICKETS; i ++){   // 2일권부터 10일권까지
```

① i일 전까지 사용

② i일째 최적값

③ i+1 이후부터 최적값 계산

④ 결과 출력

```
            setPrice=price[i];              // i일권 가격을 setPrice에 저장하기
            if(minPrice[i] > setPrice)      // 이전 단계(i-1일권까지만 이용한)에서 구한 최소 비용보다
                minPrice[i]=setPrice;       // i일권을 사용하는 것이 더 저렴한 경우 "최소 비용" 배열 값 갱신
                                            // i일차까지의 최소 비용 확정됨
            for(j =i +1; j <=days; j ++)
                if(minPrice[j]> (setPrice +minPrice[j -i]) )
                    minPrice[j]=(setPrice +minPrice[j -i]);
            for(k =1; k <=days; k ++)  // i일권까지 사용했을 때의 일별 최소 비용 출력
                printf("%3d  ", minPrice[k]);
            printf("₩n₩n");
        }
        printf("₩n최소 비용: %d원₩n", minPrice[days]*1000);  // 스키장 이용일수에 대한 최소 비용 출력
        return 0;                                             // 단위가 1000원이므로 1000 곱해서 출력
}
```

<실행 결과>

▸입력

▸출력