자료구조 7주차 과제

제출일: 2022-10-22
학번/이름: 2016121150 / 윤준영

문1) 다음은 후위 표기법의 적법성 검사 함수이다. 이를 전위 표기법의 적법성 검사 프로그램으로 바꾸고, 고친 함수가 항상 제대로 동작함(전위식 이면/아니면 항상 1/0 반환)을 증명하시오.(3점)

주어진 is_legal함수를 수정하여 사용하기 위해서, 중간의 if문을 f>1로 수정하면 전위 표기법 적법성 검사를 할 수 있다. 하지만 문제가 발생하는데, A+B라는 식이 들어왔을 경우에 f는 0 1 0 1 순서로 변하기 때문에 수정한 함수를 실행하면 1을 반환하게 된다. 이를 해결하기 위하여, 먼저 처음 들어오는 문자가 operator인지 확인하여야 한다. f를 -1로 초기화한 후, 첫 문자가 operator가 아닐 시 0을 반환하는 식을 추가하였다.

```
23  int is_pre_legal(char *s) {
24      int f = -1;
25      while (*s == ' ') s++;
26      if (!is_operator(*s)) return 0; // when first is not op: false
27      s++;
28      while (*s) {
29          while (*s == ' ') s++;
30          if (is_operator(*s)) f--;
31          else f++;
32          if (f > 1) break; // check situation like A+B
33          s++;
34      }
35      return (f == 1); // legal if operand-operator==1
36  }
```

전위 표기법의 경우 <op><exp><exp>로 구성되어 있다.
옳은 전위 표기법이 들어왔을 경우, f는 <op><exp><exp>에서 첫 문자가 op인지 검사한 후 f=-1로 시작하여 <exp><exp>를 거치게 된다. 첫 <exp>가 숫자 하나라면, 바로 f=0으로 증가하여 두 번째 <exp>를 거치게 된다. 첫 <exp>가 옳은 전위표기법일 경우에도 f=0으로 증가하여 두 번째 <exp>를 거치게 되고, 마찬가지로 두 번째 <exp>에서도 옳을 경우 f=1로 증가하여 f==1에 의하여 1을 반환하게 된다.
만약 틀린 전위 표기법일 경우, 먼저 첫 문자가 operator인지 확인하고, 아니면 0을 반환한다. 그 후 주어진 operator의 개수에 해당하는 숫자 또는 <exp>를 초과하여 들어왔을 경우 f>1의 조건문에 의하여 false(0)를, 숫자 또는 <exp>가 해당 개수가 되지 않을 경우에도 f==1 조건문에 의하여 false(0)를 반환하게 된다.

문2) 막대기의 길이에 따른 가격이 다음과 같다. (길이가 11이상이면 팔 수 없다.)

price[ ]

0	1	2	3	4	5	6	7	8	9	10
0	1	5	8	9	10	17	17	20	22	26

주어진 길이의 막대기에 대하여 가장 비싸게 자른 가격을 계산하는 프로그램을 작성하려고 한다. 길이가 m인 막대기의 가장 비싼 가격을 maxp(m)이라고 했을 때 maxp(m)은 다음과 같이 재귀적으로 계산될 수 있다. (maxp(0) = 0)

maxp(0) = 0

0<m<10 일 경우

maxp(m) = MAX{ price[1]+maxp(m−1), price[2]+maxp(m−2), ... ,
price[m−1]+maxp(1), price[m]+maxp(0) }

m>=10 일 경우

maxp(m) = MAX{ price[1]+maxp(m−1), price[2]+maxp(m−2), ... ,
price[9]+maxp(m−9), price[10]+maxp(m−10) }

1) 위 방법을 사용하여 maxp 함수를 사용하는 다음 프로그램을 완성하시오.(3점)

HW > HW7 > C hw7_2.c > ...

```c
#include <stdio.h>

int price[] = {0,1,5,8,9,10,17,17,20,22,26};
int maxp(int m) {
    int i,p;
    int max; //최대 가격
    int maxCut = 10;
    if (m<=0) return 0;
    if (m<=10) maxCut = m;
    max = 0;
    for (i=1; i<=maxCut; i++) {
        p = price[i] + maxp(m-i);
        if (p > max) max = p;
    }
    return max;
}

int main() {
    int l;
    printf("Length of stick? ");
    scanf("%d", &l);
    int p = maxp(l);
    printf("Maximum price:%d\n",p);
    return 0;
}
```

2) 위 프로그램은 m이 증가함에 따라(예: m:20 --> m:30) 급격하게 시간이 오래 걸린다. 그 이유를 설명하시오.(1점)\
maxp(m)를 계산할 때, price[i]+maxp(m-i)를 i=1~10까지 10회 계산하게 된다. 각 반복만다 maxp(m-i)를 이용하여 다시 계산하게 되는데, maxp(m)을 계산할 때, maxp(m-1) ~ maxp(m-10)을 10회 계산하여야 하고, 다시 각 항마다 maxp(m-2)~maxp(m-11)부터 maxp(m-11) ~ maxp(m-20)을 다시 계산하여야 하므로, 계산이 기하급수적으로 늘어나게 된다.
여기서 T(n)을 계산하게 되면,
T(n) = 10+T(n-1)+T(n-2)+...+T(n-10)
= 2*{10+T(n-2)+T(n-3)+...+T(n-10)}+T(n-11)
= 4*{10+T(n-3)+T(n-4)+...+T(n-10)}+3*T(n-11)+2*T(n-12)
= 8*{10+T(n-4)+T(n-5)+...+T(n-10)}+7*T(n-11)+6*T(n-12)+4*T(n-13)
~ O(2^n) 이 되게 되므로 m=20에서 30으로 증가하게 되면 기하급수적으로 증가하게 된다.

3) 동적 프로그래밍 기법을 사용하여 2번 프로그램의 성능을 개선하고자 한다. fibo 함수에서처럼 중간 결과를 저장함으로써, 성능을 높일 수 있다. 전역 배열에 이미 계산되었던 값을 저장하여 성능을 높이도록 maxp 함수를 수정하시오. (3점)

```
5  int maxprices[100] = {0}; /* 최대 가격을 일단 0으로 초기화 */
6  int price[] = {0,1,5,8,9,10,17,17,20,22,26};
7  int maxp(int m) {
8     if (maxprices[m]!=0) return maxprices[m];
9     int i,p;
10    int max;
11    int maxCut = 10;
12    if (m<=0) return 0;
13    if (m<=10) maxCut = m;
14    max = 0;
15    for (i=1; i<=maxCut; i++) {
16       p = price[i] + maxp(m-i);
17       if (p > max) max = p;
18    }
19    maxprices[m] = max;
20    return max;
21 }
```

19줄을 보면, max값을 반환하기 전에 maxprices 행렬에 결과를 미리 저장하는 명령을 추가했다. 8줄을 보면, maxprices[m]의 값이 0이 아닐 경우(이미 존재할 경우) 아래 함수 명령을 실행하지 않고 바로 그 값을 반환하게 한다. 따라서 위 재귀함수의 특성상 같은 maxp연산을 반복하는데 이 때의 시간을 줄일 수 있다. 동적 프로그래밍 기법을 사용하지 않았을 때와 사용하였을 때의 시간을 계산해 본 결과는 우측과 같다.
계산 결과 length를 30으로 하였을 때, 동적 프로그래밍을 사용하지 않았을 때에는 12s정도 소요되었지만, 사용하였을 때에는 1μs미만인 것을 확인할 수 있었다.

```
Length of stick? 30
Maximum price:85
time: 11.904786
Maximum price(dynamic):85
time(dynamic): 0.000000
```