자료구조와실습
연습문제

학번 : 2016110056
학과 : 불교학부
이름 : 박승원
날짜 : 2016년 9월 25일

1. 다음 중 추상 자료형의 설명 중 틀린 것은?

   (a) 추상자료형은 구현의 세부적인 사항을 무시한다.

   (b) 자료구조의 구현이 바뀌더라도 추상 자료형의 연산만을 사용하였다면 응용 프로그램을 바꾸지 않아도 된다.

   (c) 추상자료형을 사용하면 프로그램의 수행속도가 빨라진다.

   (d) 자세하고 명확한 인터페이스를 사용함으로써 오류의 가능성을 줄인다.

2. Set(집합) 추상 데이터 타입을 정의하라. 다음과 같은 연산자들을 포함시켜라.

   `Create, insert, remove, is_in, union, intersection, difference`

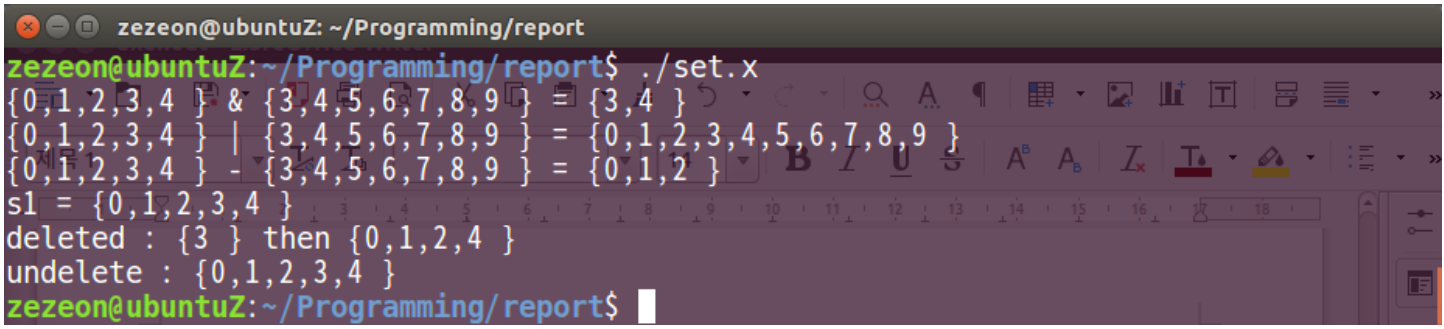
3. Boolean 추상 데이터 타입을 정의하고 다음과 같은 연산자들을 포함시켜라.

   And or not xor

```
#include<iostream>
using namespace std;

template <typename T, int N> class Set
{
public:
   bool on[N+1] {};//true일 때만 그에 대응하는 배열의 요소가 유효.
   T arr[N+1];//데이터를 담을 배열, 1 인덱스부터 유효.

   int is_in(T n) const{//return 0 if none
      for(int i=1; i<=N; i++) if(on[i] && arr[i] == n) return i;
      return 0;
   }

   void insert(T n) {
      if(!is_in(n)) {
         for(int i=1; i<=N; i++) {
            if(!on[i]) {
               arr[i] = n;
               on[i] = true;
               break;
            }
         }
      }
   }

   void remove(T n) {
      on[is_in(n)] = false;
   }


   template <int N2>
   Set<T, N> operator&(const Set<T, N2>& r) {
      Set<T, N> s;
```

```
    for(int i=1; i<=N; i++) if(on[i] && r.is_in(arr[i])) s.insert(arr[i]);
    return s;
}

template <int N2>
Set<T, N+N2> operator|(const Set<T, N2>& r) {
    Set<T, N+N2> s;
    for(int i=1; i<=N; i++) if(on[i]) s.insert(arr[i]);
    for(int i=1; i<=N2; i++) if(r.on[i]) s.insert(r.arr[i]);
    return s;
}

template <int N2>
Set<T, N> operator-(const Set<T, N2>& r) {
    Set<T, N> s;
    for(int i=1; i<=N; i++) if(on[i]) s.insert(arr[i]);
    for(int i=1; i<=N2; i++) if(r.on[i]) s.remove(r.arr[i]);
    return s;
}

friend ostream& operator<<(ostream& o, const Set<T, N>& r) {// const 있어야 함.
    o << "{ ";
    for(int i=1; i<=N; i++) if(r.on[i]) o << r.arr[i] << ',';
    o << "\b }";
    return o;
}

Set<T, N> operator!() {
    Set<T, N> s;
    for(int i=1; i<=N; i++) s.on[i] = !on[i];
    return s;
}

Set<T, N> operator&&(const Set<T, N>& r) {
    Set<T, N> s;
    for(int i=1; i<=N; i++) s.on[i] = on[i] && r.on[i];
    return s;
}
Set<T, N> operator||(const Set<T, N>& r) {
    Set<T, N> s;
    for(int i=1; i<=N; i++) s.on[i] = on[i] || r.on[i];
    return s;
}
Set<T, N> operator^(const Set<T, N>& r) {
    Set<T, N> s;
    for(int i=1; i<=N; i++) s.on[i] = !(on[i] == r.on[i]);
    return s;
}
```

```
protected:
};


int main()
{
   Set<int, 5> s1;
   for(int i=0; i<5; i++) s1.insert(i);
   Set<int, 10> s2;
   for(int i=3; i<10; i++) s2.insert(i);
   cout << s1 << " & " << s2 << " = " << (s1 & s2) << endl;
   cout << s1 << " | " << s2 << " = " << (s1 | s2) << endl;
   cout << s1 << " - " << s2 << " = " << (s1 - s2) << endl;
   cout << "s1 = " << s1 << endl;
   s1.remove(3);
   cout << "deleted : " << !s1 << " then " << s1 << endl;
   cout << "undelete : " << (!s1 || s1) << endl;

}
```




```
//in C language
#include<stdio.h>
#define MAX 100
typedef int element;
typedef int boolean;
typedef struct Set_Struct {
   element array[MAX];
   boolean valid[MAX];
} Set;

void init(Set* A)
{
   for(int i=0; i<MAX; i++) A->valid[i] = 0;
}

int is_in(Set* A, element n)
{
   for(int i=0; i<MAX; i++) if(n == A->array[i] && A->valid[i]) return 1;
   return 0;
```

```
}

void remov(Set* A, element n)
{
   if(is_in(A, n))
      for(int i=0; i<MAX; i++) if(n == A->array[i] && A->valid[i]) A->valid[i] = 0;
}

void insert(Set* A, element n)
{
   if(!is_in(A, n)) {
      for(int i=0; i<MAX; i++) {
         if(!A->valid[i]) {
            A->array[i] = n;
            A->valid[i] = 1;
            break;
         }
      }
   }
}

Set union_sets(Set A, Set B)
{
   Set C;
   init(&C);
   for(int i=0; i<MAX; i++) if(A.valid[i]) insert(&C, A.array[i]);
   for(int i=0; i<MAX; i++) if(B.valid[i]) insert(&C, B.array[i]);
   return C;
}

Set inter_sets(Set A, Set B)
{
   Set C;
   init(&C);
   for(int i=0; i<MAX; i++)
      if(A.valid[i] && is_in(&B, A.array[i])) insert(&C, A.array[i]);
   return C;
}

Set minus_sets(Set A, Set B)
{
   Set C;
   init(&C);
   for(int i=0; i<MAX; i++) if(A.valid[i]) insert(&C, A.array[i]);
   for(int i=0; i<MAX; i++) if(B.valid[i]) remov(&C, B.array[i]);
   return C;
}
```

```
void show(Set* A)
{
   printf("{ ");
   for(int i=0; i<MAX; i++) if(A->valid[i]) printf("%d,", A->array[i]);
   printf("\b}");
}

Set and(Set A, Set B)
{
   Set C;
   for(int i=0; i<MAX; i++) C.valid[i] = A.valid[i] && B.valid[i];
   return C;
}

Set or(Set A, Set B)
{
   Set C;
   for(int i=0; i<MAX; i++) C.valid[i] = A.valid[i] || B.valid[i];
   return C;
}

Set not(Set A)
{
   Set C;
   for(int i=0; i<MAX; i++) C.array[i] = A.array[i];
   for(int i=0; i<MAX; i++) C.valid[i] = !A.valid[i];
   return C;
}

Set xor(Set A, Set B)
{
   Set C;
   for(int i=0; i<MAX; i++) C.valid[i] = !(A.valid[i] == B.valid[i]);
   return C;
}

int main()
{
   Set A,B,C;
   init(&A); init(&B); init(&C);
   for(int i=0; i<5; i++) insert(&A, i);
   for(int i=0; i<5; i++) insert(&B, i+2);
   show(&A);
   show(&B);
   C = union_sets(A, B);
   show(&C);
   C = inter_sets(A, B);
   show(&C);
```

```
}
```

4. $n^2 + 10n + 8$의 시간 복잡도 함수를 빅오 표기법으로 나타내면 ?

   (a) O(n)

   (b) O($log_2 n$)

   (c) **O(n^2)**

   (d) O($n^2 log_2 n$)

5. 시간복잡도 함수가 이라면 이것이 나타내는 거승ㄴ 무엇인가?

   (a) **연산의 회수**

   (b) 프로그램의 수행시간

   (c) 프로그램이 차지하는 메모리의 양

   (d) 입력 데이터의 총개수

6. O(n^2)의 시간복잡도를 가지는 알고리즘에서 입력의 개수가 2배로 되었다면 실행시간은 어떻게 되는가?

   (a) 변함없다.

   (b) 2배

   (c) **4배**

   (d) 8배

7. O(n^2)의 시간복잡도를 가지는 알고리즘이 1초에 입력 100을 처리한다. 이 알고리즘이 100초에 처리할 수 있는 입력의 개수는?
   $c \times 100^2 = 1$
   $c = 1/10000$
   $1/10000n^2 = 100$
   $n = 1000$

8. 다음의 빅오 표기법들을 수행시간이 적게 걸리는 것부터 나열하라.

   $O(1)$ $O(logn)$ $O(n)$ $O(nlogn)$ $O(n^2)$ $O(2^n)$ $O(n!)$

9. 다음의 코드에서 정확한 대입연산, 곱셈연산, 덧셈연산, 비교연산의 개수를 계산하여 정확한 시간 복잡도 함수 값을 계산하다.

   (a)
```
test(int n)
{
    int n;
    int total=1;//1
    for(i=2; i<n; i++) total *=n;//2(n-2)
    return n;//total = 2n-3
}
```

(b)
```
float sum(float llist[], int n)
{
   float tempsum;
   int i;
   tempsum = 0;//1
   for(i=0;i<n; i++) {//2n
      tempsum += list[i];//n
   }
   tempsum += 100;//1
   tempsum += 200;//1
   return tempsum;//total = 3n+3
}
```

(c)
```
void sum(int n)
{
   int i,b;
   b=2;//1
   i=1;//1
   while(i<=n) {//logn
      i = i*b;//logn
   }
}//total = 2logn+2
```

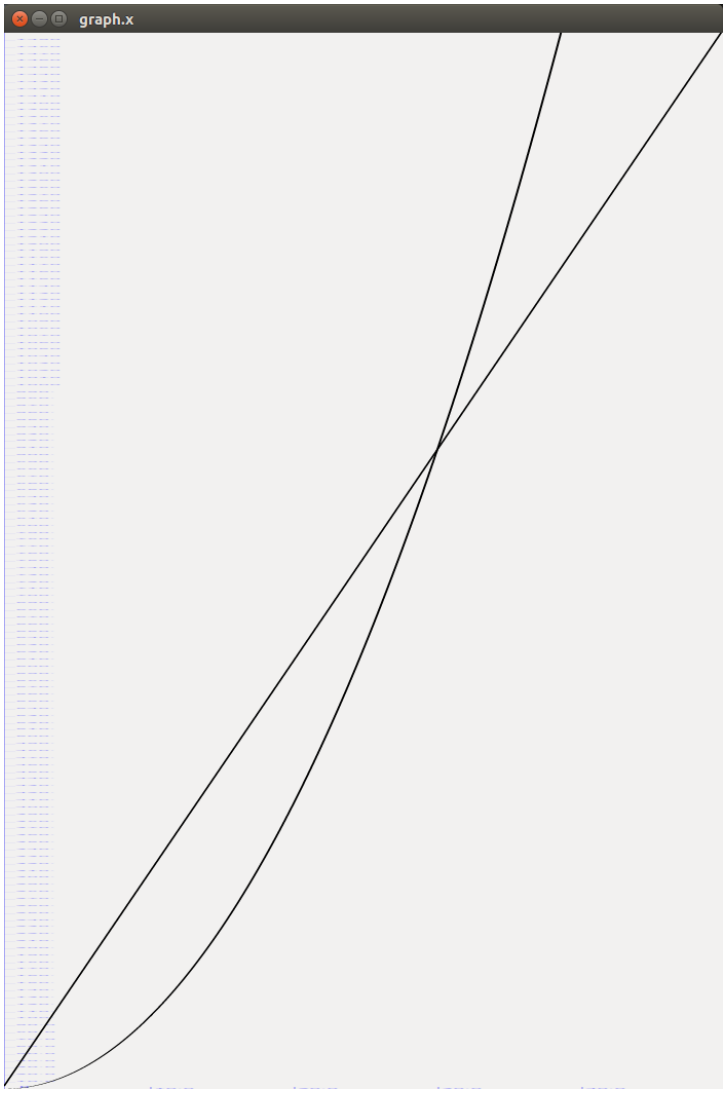
$i = 2^n$

$\therefore i = log_2 n$

10. 두 개의 알고리즘 A와 B가 있다. A의 시간 복잡도 함수는 $1000n^2 + 1000$이고 B의 시간 복잡도 함수는 2^n이라고 하자. n의 값이 어느 정도 이상이어야 A가 유리한가? 19

```
#include<iostream>
#include<cmath>
using namespace std;

int main()
{
   for(int n=0; ; n++) {
      if(1000*n*n+1000 < pow(2, n)) {
         cout << n; break;
      }
   }
}
```

11. 두함수 $30n+4$와 n^2를 여러가지 n값으로 비료하라. 언제 30n+4가 n^2보다 작은 값을 갖는지를 구하라. 그래프를 그려보라.



$n^2 - 30n + 4 = 0$

$n = 15 \pm \sqrt{229}$

12. a,b >1 인 경우에 $O(log_a n) = O(log_b n)$임을 증명하라.
$O(log_a n = \frac{logn}{loga} = \frac{1}{loga} logn = logn = \frac{1}{logb} logn = \frac{logn}{logb} = log_b n)$

13. 다음은 실제로 프로그램의 수행시간을 측정하여 도표로 나타낸 것이다. 도표로부터 이 프로그램의 시간 복잡도를 예측하여 빅오표기법으로 나타내라.

입력의 개수 n	수행시간(초)
2	2
4	8
8	25
16	63
32	162

$nlog_2 n$

14. 다음 프로그램의 시간 복잡도를 빅오표기법으로 나타내라.

(a) for(i=0; i<n; ++i) ++k; //3n=O(n)

(b) for(i=1; i<n; i*=2) ++k; $//2log_2 n = O(logn)$

(c) for(i=n-1; i != 0; i/=2) ++k;//$log_2 n = O(logn)$

(d) for(i=0; i<n; ++i) if(i%2 == 0) ++k;//$\frac{7}{2}n = O(n)$

(e) for(i=0; i<n; ++i) for(j=0; j<n; ++j) ++k;//O(n^2)

(f) for(i=0; i<n; ++i) for(j=i; j<n; ++j) ++k//$\frac{n(n+1)}{2} = O(n^2)$

(g) for(i=0; i<n; ++i) for(j=0; j<n; ++j) for(r=0; r<10; ++r) ++k;//$10n^2 = O(n^2)$

15. sub함수의 시간 복잡도가 O(n)일 때 다음 문장의 시간 복잡도는?

for(i=0; i<n; i*=2) sub();//O(nlogn)

16. 빅오 표기법의 정의를 사용하여 다음을 증명하라.

(a) $5n^2 + 3 = O(n^2)$ $\quad \because \lim_{n\to\infty} \frac{5n^2+3}{n^2} = 5$

(b) $wn^2 + 10n = O(n^2)$ $\quad \because \lim_{n\to\infty} \frac{n^2+10n}{n^2} = 1$

(c) $7n^3 + 10n^2 - 3n + 5 = O(n^3)$ $\quad \because \lim_{n\to\infty} \frac{7n^3+10n^2-3n+5}{n^3} = 7$

(d) $5n^2 + 2^n = O(2^n)$ $\quad \because \lim_{n\to\infty} \frac{5n^2+2^n}{2^n} = 1$

17. 빅오표기법의 정의를 이용하여 $6n^2 + 3n$이 O(n)이 될 수 없음을 보여라.

$\because \lim_{n\to\infty} \frac{6n^2+3n}{n} = \infty$

18. 다음의 프로그램 코드에 대하여 답하라.

```
int i, k;
for(i=0; i<(n-2); i++) {//2(n-2)
   for(k=0; k<30; k++) {//60(n-2)
      buffer[i][k] = 0;//30(n-2)
   }
}
```

(a) 다음 알고리즘의 시간 복잡도를 n에 대한 함수로 나타내고, 빅오 표기법으로도 나타내어라. 여기서 입력의 계수는 양의 정수 n이다.

O(n)

(b) 위의 프로그램에서 입력의 개수도 100배 증가하였고, CPU의 속도도 100배 증가하였을 경우, 위의 프로그램의 수행 시간이 늘어나는가? 아니면 줄어드는가? 그 이유는?

동일하다. O(n)이 일차함수이기에 입력의 개수에 정비례하므로 .

19. 다음 알고리즘의 시간 복잡도를 n에 대한 함수로 나타내고, 빅오 표기법으로도 나타내어라.

```
answer = 1.0;
temp = a;
k = n;
while (k>0) {
   if((k%2) != 0) answer *= temp;
   k = (int) k/2;//logn/log2
}
```

$3log_2 n = O(logn)$

20. 배열에 정수가 들어 있다고 가정하고 다음의 작업의 최악, 최선의 시간 복잡도를 빅오 표기법으로 말하라.

    (a) 배열의 n번째 숫자를 화면에 출력한다.
    최선 : O(1), 최악 O(1)

    (b) 배열안의 숫자 중에서 최소값을 찾는다.
    최선 : O(n), 최악 : O(n)

    (c) 배열의 모든 숫자를 더한다.
    최선 : O(n), 최악 : O(n)

21. 1시간에 10개의 입력을 처리할 수 있는 프로그램이 있다. 만일 속도가 100배 빠른 컴퓨터를 구입하여 동일한 작업을 한다면 프로그램의 시간 복잡도가 다음과 같은 경우, 1시간에 처리할 수 있는 입력의 개수는 얼마가 되겠는가?

    (a) T(n)=n
    $10 \times 100 = 1000$

    (b) T(n)=$nlog_{10}n$
    $c10log_{10}10 = 1$
    $\therefore c = 1/10$
    $1/10nlog_{10}n = 100$
    $n = 100$

    (c) T(n)=n^2
    $c \times 100 = 1$
    $\therefore c = 1/100$
    $1/100 \times n^2 = 100$
    $n = 100$

    (d) T(n)=n^3
    $c \times 1000 = 1$
    $\therefore c = 1/1000$
    $1/1000n^3 = 100$
    $n = 100000^{\frac{1}{3}}$

    (e) T(n)=10^2
    100

22. (a) 크기가 n인 배열 array에서 임의의 위치 loc에 있는 정수 value를 삽입하는 함수를 작성하라. 정수가 삽입되면 그 뒤에 있는 정수들은 한 칸씩 뒤로 밀려야 한다. 현재 배열에 들어있는 원소의 개수는 items개라고 하자. (여기서 items <n라고 가정)

```
void insert_array(int loc, int value) {
   int array[n];
   int items;
   for(int i=items-1; i>=loc; i--) {
      array[i+1] = array[i];
   }
```

```
    array[loc] = value;
    items++;
}
```

(b) 위의 연산의 최악, 최선, 평균적인 경우의 시간복잡도는?(빅오 표기법으로)
최악: O(n), 최선 : O(1), 평균 : O(n)

23. C언어의 typedef을 이용하여 complex라고 하는 새로운 데이터 타입을 정의하라. complex 데이터 타입은 구조체로서 float형인 real변수와 float형인 imaginary변수를 갖는다.

```
typedef struct {
    float real, imaginary;
} complex;
```

소감 굉장히 많은 분량이었다. 2,3번 문제를 합쳐서 풀었다. 추상데이터 타입을 정의하는 데에 기왕이면 템플레이트를 정의하는 것도 손에 익힐 겸, 범용으로 만들어 보았다. typename, class며 keyword도 많고, template은 굉장히 구문이 난삽해서 항상 헷갈린다. 자주 사용해서 손에 익히는 수밖에 없을 것 같다. 16번은 교수님께 여쭤본 방법으로 풀었다. 나눠서 n을 무한대로 보낼 때에 0 혹은 무한대가 되지 않는 계수가 없는 가장 간단한 함수로 빅오를 이해할 수 있다.