자료구조와실습
연습문제

학번 : 2016110056
학과 : 불교학부
이름 : 박승원
날짜 : 2016년 9월 23일

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] {};//true일 때만 그에 대응하는 배열의 요소가 유효.
   T arr[N];//데이터를 담을 배열

   void insert(T n) {
      if(!is_in(n)) {
         for(int i=0; i<N; i++) {
            if(!on[i]) {
               arr[i] = n;
               on[i] = true;
               break;
            }
         }
      }
   }

   void remove(T n) {
      for(int i=0; i<N; i++) if(on[i] && arr[i] == n) on[i] = false;
   }

   bool is_in(T n) {
      for(int i=0; i<N; i++) if(on[i] && arr[i] == n) return true;
      return false;
   }

   template <int N2>
   Set<T, N> operator&(const Set<T, N2>& r) {
      Set<T, N> s;
      for(int i=0; i<N; i++)
```

```
      if(on[i]) for(int j=0; j<N2; j++)
         if(r.on[j] && arr[i] == r.arr[j]) 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=0; i<N; i++) if(on[i]) s.insert(arr[i]);
   for(int i=0; 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=0; i<N; i++) if(on[i]) s.insert(arr[i]);
   for(int i=0; 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=0; 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=0; 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=0; 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=0; 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=0; 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;

}
```

```
zezeon@ubuntuZ: ~/Programming/report
zezeon@ubuntuZ:~/Programming/report$ ./set.x
{0,1,2,3,4 } & {3,4,5,6,7,8,9 } = {3,4 }
{0,1,2,3,4 } | {3,4,5,6,7,8,9 } = {0,1,2,3,4,5,6,7,8,9 }
{0,1,2,3,4 } - {3,4,5,6,7,8,9 } = {0,1,2 }
s1 = {0,1,2,3,4 }
deleted : {3 } then {0,1,2,4 }
undelete : {0,1,2,3,4 }
zezeon@ubuntuZ:~/Programming/report$
```

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?
10000

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
```

$i = 2^n$

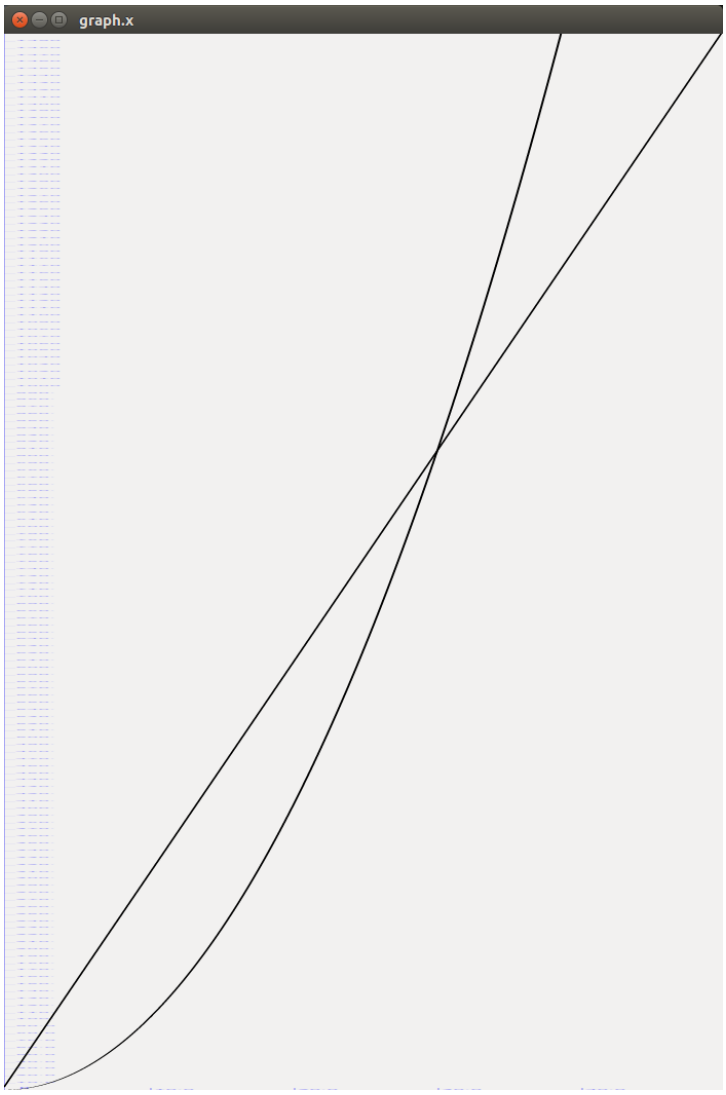
$i = log_2 n$

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

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

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

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



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

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