1.1 자료구조와 알고리즘

자료구조란? 자료들을 정리하여 보관하는 여러가지 구조

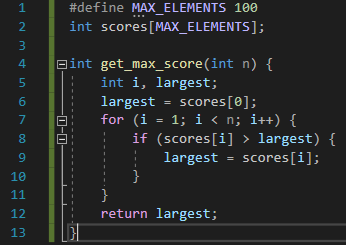
스택 자료들을 쌓아서 정리하는 구조

맨 위에서만 자료 추가∙제거

큐 먼저 도착한 자료가 먼저 빠져나가는 자료구조

프로그램 = 자료구조 + 알고리즘 ex) 시험성적을 읽어들여서 최고 성적을 구하는 프로그램 자료구조 – 배열, 알고리즘 – 최댓값 비교

프로그램1.1



알고리즘이란? 컴퓨터를 통한 문제해결 1. 문제를 해결하는 방법고안

2. 컴퓨터가 수행할 절차를 자세히 기술

알고리즘 컴퓨터로 문제를 풀기위한 단계적인 절차

문제와 컴퓨터가 주어진 상태에서, 문제를 해결하는 방법을 정밀하게 장치가 이해할 수 있는 언어로 기술한 것

정의1.1 알고리즘 입력 0개 이상의 입력이 존재하여야 한다

출력 1개 이상의 출력이 존재하여야 한다

명백성 각 명령어의 의미는 모호하지 않고 명확해야 한다

유한성 한정된 수의 단계 후에는 반드시 종료되어야 한다

유효성 각 명령어들은 종이와 연필 또는 컴퓨터로 실행가능한 연산이어야 한다

알고리즘 기술 시, 의사코드∙프로그래밍 언어가 많이 쓰임

Quiz 1. 알고리즘

2. 의사코드

3. 4

1.2 추상 자료형

자료형 데이터의 종류

ex) 정수, 실수, 문자열, 큐, 스택, 리스트, 트리

C언어 제공 자료형 기초자료형 char, int, float, double

파생자료형 배열, 포인터

사용자 정의 자료형 구조체, 공용체, 열거형

자료형 작성 데이터(컴퓨터에서 나타낼 수 있는 데이터), 연산(지료형 간에 가능한 연산) 고려

추상자료형 Abstract Data Type

추상적, 수학적으로 자료형을 정의한 것 = 실제적인 구현으로부터 분리되어 정의된 자료형 = 자료형을 추상적으로 정의한 것

데이터, 연산이 무엇인지는 정의하지만, 연산을 어떻게 구현할 것인지는 정의 X

객체, 함수 정의 객체 데이터 정의 ex) 0부터 시작해 INT\_MAX까지의 순서화된 정수의 부분범위

함수 연산 정의 ex) Boolean is\_zero(x) :: = if (x) return False //0인지 판별하는 연산

else return True

함수의 이름, 함수의 매개변수, 함수의 반환형, 함수 기능에 대한 기술

::= ~으로 정의된다

클래스

구현으로부터 명세의 분리 ADT의 인터페이스만 정확하게 지켜진다면, 다른 방법으로 구현해도 됨

자료구조 추상자료형을 프로그래밍 언어로 구현한 것

좋은 추상화 핵심적인 구조, 동작에만 집중

Quiz 1. 함수

2. 추상

3. Boolean is\_greater(x, y) :: = if (x>y) return True

else return False

1.3 자료구조와 알고리즘

상용 프로그램의 규모가 커짐 🡪 처리해야할 자료의 양이 많아짐 🡪 알고리즘의 효율성 중요

사용자들의 빠른 프로그램 선호

수행시간 측정방법 알고리즘을 프로그래밍언어로 작성하여 실제 프로그램상에서 샐행시킨 다음, 수행시간을 측정 🡪 정확, 확실

문제 하드웨어에 따라 다르게 측정될 수 있음

방법1 #include <stdio.h>

#include <ctime.h>

start = clock(); //clock(): 호출 프로세스에 의해 사용된 CPU 시간 계산

//clock(): CLOCK\_PER\_SEC 단위로 반환

//알고리즘을 시작하기 전에 clock()을 호출해 start변수에 저장

...

stop = clock(); //알고리즘이 끝나면 clock()을 호출해 stop변수에 저장

double duration = (double)(stop = start) / CLOCKS\_PER\_SEC; //초단위 시간 측정

방법2 #include <time.h>

start = time(NULL); //time()함수는 초 단위 측정된 시간 반환

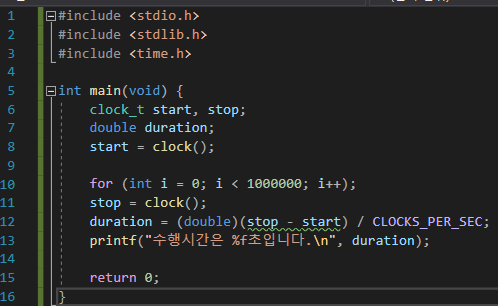
//time(NULL) 형태로 호출하면 현재 시간이 넘어옴

...

stop = time(NULL);

double duration = (double)difftime(stop, start); //두 가지 시간을 difftime()에 인수로 넣으면 차이가 초단위로 반환

프로그램1.2



알고리즘의 복잡도 분석방법 알고리즘의 효율성을 따져보는 기법들

알고리즘 복잡도 분석 직접 구현하지않고도 대략적으로 알고리즘의 효율성을 비교

시간 복잡도 함수 시간복잡도 알고리즘의 수행시간 분석

알고리즘을 이루고 있는 연산들이 몇 번이나 수행되는 지 숫자로 표시

입력에 따라 수행되는 연산의 수행횟수가 달라짐

공간복잡도 알고리즘이 사용하는 기억공간 분석

빅오표기법 입력의 개수(n)과 시간복잡도함수(T(n))의 관계 자료의 개수가 많은 경우, 차수가 가장 큰항이 가장 큰 영향을 미침, 다른 항들은 상대적으로 무시 가능

보통 시간복잡도 함수에서 차수가 가장 큰 항만 고려

중요한 것은 입력값이 증가함에 따른 증가 추세

시간복잡도 함수에서 불필요한 정보를 제거해 알고리즘 분석을 쉽게 할 목적으로 시간복잡도를 표시하는 방법

빅오 of n O(n)

n의 값에 따른 함수의 상한 값을 나타내는 방법 ex) f(n) = n, O(n) f(n) = 5n2+3n, O(n2)

정의1.2 빅오 표기법 두개의 함수 f(n)과 g(n)이 주어졌을 때 모든 n>n0에 대하여 |f(n)| <= c|g(n)|을 만족하는 2개의 상수 c와 n0가 존재하면 f(n)=O(g(n))이다.

빅오표기법을 사용하면 시간복잡도를 간단히 표시 가능

n이 작을 때는, 상수항이나 각항도 영향을 끼침

지수형, 팩토리얼형의 빅오 of n을 가지는 경우 사용 불가능

빅오표기법 이외의 표기법 빅오표기법의 문제 상한이 여러 개 존재할 수 있음

빅오메가 어떤 함수의 하한을 표시하는 방법

정의1.3 빅오메가 표기법 f(n)과 g(n)이 주어졌을 때 모든 n>n0에 대하여 |f(n)|>=c|g(n)|을 만족하는 두개의 상수 c, n0이 존재하면 f(n) = Ω(g(n))이다

빅세타 동일한 함수로 상한과 하한을 만들수 있는 경우, f(n) = O(g(n))이고 f(n) = Ω(g(n))인 경우를 f(n) = Θ(g(n))이라 한다

정의1.4 빅세타표기법 f(n)과 g(n)이 주어졌을 때 n>n0에 대하여 c1|g(n)| <= |f(n)| <=c2|g(n)|을 만족하는 상수 c1, c2, n0이 존재하면 f(n) = Θ(g(n))이다

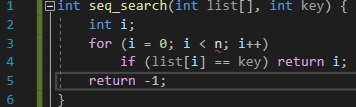
가장 정밀 but 통상적으로 빅오표기법 많이 사용

최선, 평균, 최악의 경우 입력의 집합에 따라 다른 수행시간을 보일 수 있음 🡪 자료집합의 기준 필요

최악의 경우 자료집합 중에서 알고리즘의 수행시간이 가장 오래 걸리는 경우

알고리즘의 시간 복잡도 척도로 많이 쓰임

최선의 경우 자료집합 중에서 알고리즘의 수행시간이 가장 적게 걸리는 경우

프로그램1.3 

찾고자 하는 숫자가 맨 앞에 있는 경우 O(1)

찾고자 하는 숫자가 맨 뒤에 있는 경우 O(n)

평균적인 경우 (n+1)/2 = O(n)

Quiz 1. (1) O(n2)

1. (2) O(n!)

1. (3) O(n2)

1. (4) O(n2)

연습문제

1. void swap(a, b) ::= tmp=b

b=a

a=tmp

2. int big\_num(a, b) ::= if(a>b) return a

else return b

3. int factorial(n)::= sum=0

for i <-- 1 to i <--n

sum+=i

return sum

4. 객체: 자료형을 요소로 가지는 집합

함수:

set Create(a, b, c, ...) ::= set\_a = (a, b, c, ...)

return set\_a

void Insert(set\_a, b) ::= set\_a += (b)

void Remove(set\_a, b) ::= if (b in set\_a) set\_a -= b

bool Is\_In(set\_a, b) ::= return (b in set\_a)

set Union(set\_a, set\_b) ::= return set\_a + set\_b

set Intersection(set\_a, set\_b) ::= return set-a-(set\_a-set\_b)

set Difference(set\_a, set\_b) ::= return set\_a - set\_b

5. 객체: 참(1)과 거짓(0)

함수:

bool and(bl1,bl2) ::= return bl1 && bl2

bool or(bl1,bl2) ::= return bl1 || bl2

bool not(bl) ::= return ~bl

bool xor(bl1, bl2) if(b1==b2) return false

else return true

6. 2n

7. 4n2

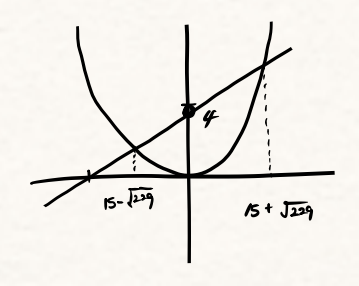
8. (3)

9. (1)

10. (3)

11. (2)

12. O(1) 🡪 O(log n) 🡪 O(nlog n) 🡪 O(n2) 🡪 O(2n) 🡪 O(n!)

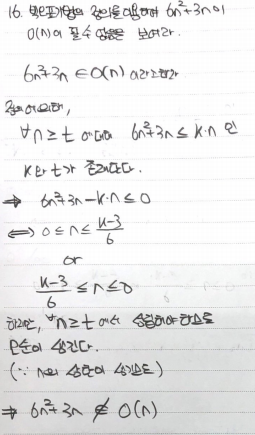
13. 0<n2-30n-4를 만족할 때, 

14. O(nlog2n)

15. 5n2+3 = O(n2)

5n2+3 < 9n2, n>1

따라서 옳음

16. 

17. (1) 최선:O(n), 최악O(1)

17. (2) 최선:O(n), 최악:O(n)

17. (3) 최선:O(n), 최악:O(n)