과제 1.

연습문제 풀이

참고사항)

* 인쇄한 후, 용지에 직접 필기하세요. 답만 적지 말고 풀이과정도 함께 기록하세요.
* 답은 필기로 쓰세요. 컴퓨터로 작성하지 마세요. 너무 오래 걸립니다.
* 이 페이지는 표지입니다. 적당히 바꿔서 사용하세요.
* 칸이 부족하면 칸에는 답만 기록하고 과정은 별지를 뒤에 추가해서 작성하세요.
* 모르는 문제는 [건너뜀] 이라고 적으세요. [건너뜀]은 전체 문제의 20% 이하여야 합니다.
* 오답이 많다고 평가가 나쁘게 되는 것은 아닙니다. 과정 없이 답만 있으면 자신이 풀지 않았다고 오해할 수 있습니다.

제출자 :

# 개본 개념

1. 공간 복잡도보다 시간 복잡도를 더 중요시 생각하는 이유는 무엇인가?
2. 이진 탐색은 빠른 탐색 방법이다. 그런데 이진 탐색이 이루어지려면 사전에 정렬이 되어 있어야 하고, 정렬의 시간까지 계산하면 이진 탐색은 결코 빠르다고 할 수 없다. 그러면 이진 탐색은 나쁜 방법인가? 만일 이진 탐색이 좋은 방법이라면 그 이유는 무엇이고, 어떻게 증명할 것인가?
3. 정렬된 1000개의 데이터에서 이진 탐색을 통해 원하는 데이터를 찾는 binsearch( ) 함수를 재귀적으로 구현하라. binsearch( ) 함수의 원형은 다음과 같다.

|  |
| --- |
| int BSearchRecur(int ar[], int first, int last, int key)  반환값 : 찾으려는 데이터가 있는 위치(인덱스)  ar : 데이터가 있는 배열  first, last : 찾으려는 배열의 범위  key : 찾으려는 값 |

1. 1에서 100 사이의 솟수를 찾는 프로그램을 작성하라. 이 프로그램의 시간적 복잡도는 어떻게 될까?

# 배열, 구조

1. rand 함수는 0부터 RAND\_MAX 사이의 숫자를 랜덤하게 생성한다. 보통 RAND\_MAX 는 32767정도의 숫자인데, 0부터 99999999 사이의 숫자를 랜덤하게 생성하려면 rand 함수만으로 해결할 수 없다. 이 문제를 해결할 방법은 무엇인지 설명하라.
2. 다음의 코드에서, number 의 시작 주소가 1000 번지라면 ptr1과 ptr2에는 어떤 숫자가 들어갈까?

|  |
| --- |
| int number[4][6];  int (\*ptr1)[6] = number;  int \*ptr2 = number[1];  ptr1++;  prr2++; |

1. 다음의 코드에서 4\*6짜리 2차원 배열을 동적 할당하여 사용하려 한다. 이 때, ptr 변수는 어떻게 선언해야 할까?

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ptr = malloc(sizeof(int) \* 4 \* 6 ) ;  for(i=0;i<4;i++)  for(j=0;j<6;j++)  ptr[i][j] = i \* j; |

1. realloc 은 동적 할당 받은 메모리의 크기를 변경하는 함수이다. 그런데, realloc을 사용할 때,

|  |  |
| --- | --- |
| ptr = malloc(10);  … | |
| 1. realloc( ptr, 100); | 1. ptr = realloc( ptr , 100); |

(1)과 (2) 중 어느 것이 올바른 방법인가? 또 그 이유는 무엇인가?

1. 다음의 코드에 오류가 있는 이유는 무엇인가?

|  |
| --- |
| struct STUDENT {  int kor;  int eng;  struct STUDENT mystd;  }; |

1. 희소 행렬은 0을 다수 포함하고 있는 행렬이며, 이것은 규격과 일치하는 크기의 배열에 저장하는 것보다 0이 아닌 값의 행과 열, 값을 저장하는 방식을 쓰는 것이 더 공간적 복잡도를 감소시키는 방법이다. 만일 규격이 M\*N인 희소 행렬에 0이 아닌 데이터가 X 개 있을 때, M\*N 규격의 배열을 쓰는 것보다 앞에서 설명한 희소 행렬 저장 방식을 쓰는 것이 더 유리한 환경인 경우는 어떤 경우인지 M, N, X와 관련된 수식으로 표현하라.

# 스택/큐

1. 다음의 중위표기법으로 된 수식을 후위표기법으로 변환하라.
2. A – B \* C + D / E – F
3. A \* ( B – ( C / D ) ) + ( E + ( F \* G ) ) / H
4. 스택은 기본적으로 PUSH와 POP 이라는 동작을 가지며, 이를 운영하기 위해 Top 이라는 변수를 안에 내장하고 있다. 우리는 새로운 스택을 설계하려고 하는데, PUSH, POP 외에 GET\_MIN(스택에 들어 있는 값들 중 최소값을 돌려줌) 이라는 동작을 하나 추가하려고 한다. 그런데 우리의 목표는 GET\_MIN 이 O(1)의 시간이 소요되도록 만들려고 한다. 이를 달성하려면 스택을 어떻게 설계해야 하는지 설명하라. O(1)의 시간이 소요된다는 의미는 스택에 있는 데이터의 개수와 무관하게 일정한 수의 명령만으로 동작을 수행할 수 있다는 뜻이다.

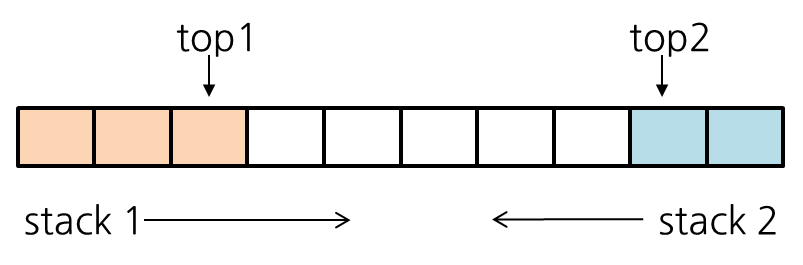
(힌트) 스택에 변수를 추가할 수도 있다.

1. 다음은 중위표기법을 후위표기법으로 바꾸는 코드이다. 그런데 여기에 기능을 추가하여

|  |
| --- |
| C = 3 + 4 \* 5 |

처럼 변수에 대입하는 기능을 넣으려고 한다. 그러면 다음의 코드의 어느 부분이 어떻게 바뀌어야 하는지 간략히 설명하라(프로그램을 작성하지 말고 설명만 하라).

1. 수식을 계산할 때, 중위 표기법을 후위 표기법으로 변환한 후, 후위 표기법을 계산하는 식이 용이하다고 한다. 만일 그 방법을 쓰지 않고 중위 표기법으로 구성된 수식을 바로 계산하는 프로그램을 작성한다면 어떤 방법을 쓰면 좋을지 설명하라. 단, 계산식의 연산자는 사칙연산 뿐이고, 괄호는 포함되지 않는다고 가정하자.
2. 10개의 배열을 이용해 만든 이중 스택이 있다. 이 스택에서 is\_full\_stack1( ) : (1번 스택이 가득 찼는지 검사하기)과 is\_full\_stack2( ) : (2번 스택이 가득 찼는지 검사하기) 는 어떻게 구현하면 좋을까?



1. 하나의 큐 대신 우선순위 큐가 포함된 2개의 큐(상위 큐, 하위 큐)를 운영하고자 한다. 상위 큐는 우선순위가 높은 큐로, 급하게 처리되어야 하는 데이터를 넣고, 하위 큐는 우선순위가 낮은 데이터를 넣어서 천천히 처리되도록 한다. DeQueue를 하는 곳에서는 상위 큐에서 2개, 하위 큐에서 1개의 비율로 데이터를 꺼내올 것이다. 이것을 구현하려면 DeQueue를 1개 할 때마다 상위 큐 : 하위 큐의 비율을 2:1로 유지해야 할 텐데, 이 구현은 어떻게 하면 좋을지 설명하라.
2. 다음은 하노이의 타워 문제를 해결하는 코드이다. 이것을 스택을 이용하여 비재귀적으로 만든다면, 스택은 어떤 모양이 되어야 할지, 프로그램은 어떻게 변경해야 할지 간단히 설명하라.

|  |
| --- |
| void hanoi(int n, char from, char by, char to)  {  if (n == 1) // 블록이 하나 뿐일 때에는 그냥 옮기면 된다.  printf("move 1 from %c to %c\n", from, to);  else {  **hanoi(n - 1, from, to, by)**;  printf("move %d from %c to %c\n", n, from, to);    **hanoi(n - 1, by, from, to);**  }  }    int main(void)  {  hanoi(8, 'A', 'C', 'B');  } |

# 정렬

1. 다음 정렬 방법들의 원소 교환 횟수가 최소인 것은 무엇인가

|  |
| --- |
| 버블 정렬, 선택 정렬, 삽입 정렬, 퀵 정렬, 힙 정렬 |

1. 퀵 정렬의 피봇을 중간에서 고른다면,
2. 어떻게 중간을 고를 수 있을까?
3. partition 함수에서 피봇보다 작은 수와 큰 수를 구분하여 왼쪽과 오른쪽에 두는데, 가운데에서 피봇을 뽑으면 기존의 partition 함수의 동작을 어떻게 수정해야 할까? 가장 쉽게 수정하는 방법은 무엇일까?
4. 다음과 같이 데이터가 있을 때(단, 배열은 1부터 시작한다고 가정하자),
5. 초기 힙을 만든 후의 상태는 어떻게 될까?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 6 | 7 | 1 | 3 | 4 | 9 | 5 | 2 |

1. 가장 큰 3개의 수를 결정한 후, 힙의 모양은 어떻게 될까? 배열의 형태로 나타내라.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

1. 데이터가 31개 있을 때, 최선의 상황에서 partition 함수는 몇 회 호출될까?

1. 힙정렬의 코드를 보자. 만일 내림차순으로 배열되려면 어느 부분이 어떻게 수정되어야 할까?

|  |  |
| --- | --- |
| void adjust(int a[], int root, int n)  {  int child, rootkey;  int temp;  temp = a[root];  rootkey = a[root];  child = 2 \* root;  while (child <= n) {  if ((child < n) &&   (a[child] < a[child + 1]))  child++;  if (rootkey > a[child])  break;  else {  a[child / 2] = a[child];  child \*= 2;  }  }  a[child / 2] = temp;  } | void heapsort(int a[], int n)  {  int i;  int temp;  for (i = n / 2; i > 0; i--)  adjust(a, i, n);  for (i = n - 1; i > 0; i--) {  SWAP(a[1], a[i + 1], temp);  adjust(a, 1, i);  }  } |

1. 힙정렬의 결과를 보면 배열의 인덱스가 1부터 N까지를 정렬하는 것을 볼 수 있다. 하지만 컴퓨터에서의 배열은 0부터 시작하기 때문에 0부터 N-1까지 동작하는 것이 더 자연스럽다고 할 수 있다. 만일 0부터 N-1까지 동작하려면 힙정렬을 어떻게 수정해야 할까?
2. 보유한 128개의 데이터가 이미 정렬되어 있는지 판단하는 방법을 설명하라.
3. 선택 정렬의 장점을 설명하라.
4. CPU는 메모리의 속도가 느린 것을 개선하기 위해 CPU 내에 캐시 메모리를 운영하면서 메모리의 일정 부분을 속도가 빠른 CPU 내의 캐시 메모리에 두고 사용한다. 그런데 힙 정렬은 이런 캐시 메모리의 운영을 방해하여 대용량 데이터의 정렬에 좋지는 않다고 한다. 퀵 정렬과 힙 정렬을 캐시메모리의 관점으로 비교하라.
5. 버블 정렬과 삽입 정렬의 성능이 상대적으로 떨어질 수밖에 없는 이유를 설명하라.
6. 퀵 정렬을 수행하는 함수(quicksort)를 작성하라. 단, quicksort 함수 내에서 quicksort 함수 이외의 다른 함수를 호출하지 않아야 한다.
7. 퀵정렬 함수는 재귀적으로 동작하여 함수 호출이 많아 성능 저하의 문제가 있다. 이를 개선하기 위해 비재귀적 방법으로 스택을 이용하는 방법을 썼는데 이것 역시 크게 성능이 개선되지는 않았다. 예상과 달리 성능 개선이 크게 되지 않은 이유는 무엇이며, 이를 해결하는 방법은 무엇일까?
8. 1에서 7 까지의 수로 이루어진 7개의 수를 퀵 정렬로 정렬할 때 최선의 경우(비교 또는 교환의 최소가 되는)가 나오는 데이터의 예를 들어라.
9. 퀵 정렬을 비재귀적 호출로 동작하게 만들려면 스택을 하나 사용해야 한다. 이 스택은 언제 PUSH를 하고 언제 POP을 하면 될까?
10. 힙정렬의 시간 복잡도가 O(nlog n) 임을 설명하라.
11. 같은 키 값을 가진 2개의 데이터가 있을 때, 정렬 후 데이터의 순서가 달라진다면 불안정한 것이고, 유지된다면 안정한 것이다. 예를 들어,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 이름 | 홍길동 | 이순신 | 이순신 | 강감찬 | 심청 |
| 전화번호 | 010-3333-4444 | 010-9382-0013 | 010-7365-9823 | 010-2211-3827 | 010-8877-8372 |

이 데이터에서 이순신은 동일한 이름을 가진 2개의 데이터가 있음을 볼 수 있다. 각각을 이순신1, 이순신2라고 하자. 두 개의 키 값은 동일하기 때문에 정렬한 후에도 그 순서가 이순신1, 이순신2로 유지될 것이다. 만일 정렬을 한 후, 이순신1, 이순신2의 순서가 유지되지 않고 바뀐다면 이 정렬은 “불안정적이다” 라고 말하고, 순서가 유지되면 “안정적이다”라고 말한다.

우리가 배운 정렬 방법 중 선택정렬, 삽입 정렬, 버블 정렬, 퀵 정렬은 안정적인 정렬 방법인지 확인하고 이를 입증할 하나의 예시 데이터를 들어라.

1. 힙 정렬에서 동일한 두 개의 키 값은 어떻게 처리되는지 확인하고 이를 통해 힙 정렬이 안정한지 불안정한지 판단하라.
2. 힙 정렬의 초기 힙을 만드는 과정에서는 데이터의 뒤에서부터 앞으로 adjust를 진행한다. 앞에서 뒤로 adjust를 하지 않고 뒤에서 앞으로 adjust를 진행하는 이유는 무엇일까? 그 사례를 하나 들어라.
3. 퀵정렬은 역순으로 정렬된 데이터를 정렬하려 할 때 최악의 상황이 생긴다고 하였다. 우리는 원본 데이터를 검토하여 이미 역순이거나 역순과 비슷한 형태로 정렬된 데이터라면 힙정렬을, 아니라면 퀵정렬을 수행하려고 한다. 이렇게 이미 역순이거나 역순에 비슷한 형태의 데이터인지 판단하는 방법은 무엇일지 설명하라.
4. 삽입 정렬에서는 비정렬 구간의 수를 골라 정렬 구간의 적절한 위치를 찾아 삽입한다고 하였다. 정렬 구간에 삽입할 적절한 위치를 찾을 때 순차 탐색이 아니라 이진 탐색을 한다면 더 성능이 개선될 것이라고 하는데, 어느 부분을 어떻게 수정하고 이를 통해 개선되는 것이 무엇인지 설명하라.
5. 퀵 정렬에서는 균형을 이룬 리스트로 분할될 때 최선의 성능을 발휘할 수 있다. 이를 위해 3개의 데이터 중 중간 값을 피봇으로 사용한 퀵 정렬이 더 효과적이라고 한다. 3개의 데이터 중 중간 값이란 (첫 번째 데이터, 가장 마지막 데이터, 정 중앙의 데이터)의 3개의 값 중 중간 값을 가져와 이를 피봇으로 이용한다는 의미이다. 만일 이렇게 동작하는 퀵정렬을 만든다면 어느 부분이 어떻게 수정되어야 할까(프로그램이 아니라 설명하기)?
6. 기수 정렬은 뒷자리부터 정렬을 하는 것이 일반적이다. 그러나 3자리의 수로 이루어진 데이터를 기수 정렬할 때, 뒷자리(1의 자리)부터 앞자리(100의 자리)의 순으로 정렬을 할 수도 있지만, 반대로 앞자리(100의 자리)부터 뒷자리로 정렬을 할 수도 있다. 두 가지 중 어느 것이 더 효율적일지 설명하라.