**C# 자료구조&알고리즘**

|  |  |  |  |
| --- | --- | --- | --- |
| 과정명 | 멀티 디바이스 메타버스 플랫폼 게임 개발자 과정 | | |
| 이름 | 임준희 | 평가일자 | 2024년 1월 26일 |

* 해당 테스트는 구현능력보다 자료구조&알고리즘에 대한 내용들에 대한 기초지식을 요구합니다.
* 문제를 풀어보고 모르는 부분에 대해서 다시 학습하기 위한 용도로 사용합니다.
* 모르는 문제가 있다면 반드시 다시 학습하는 것을 권장합니다.

1. 다음 중 알고리즘에 대한 설명으로 옳지 않은 것을 고르시오.
   1. 알고리즘이란 문제를 해결하기 위한 정해진 진행절차나 방법으로 프로그램에서 어떠한 행동을 하기 위해 만들어진 명령어 집합이다.
   2. 알고리즘은 정해진 입력에 따라 매번 다른 출력이 나올 수 있으며, 무한히 작업을 반복할 수 있으므로 입력에 대해 신중해야 한다.
   3. 알고리즘의 대략적인 효율에 대해 평가하는 점근 표기법으로 O(Big-O) 표기법이 있으며, 처리횟수 증가에 따른 증가양상을 표현하는 방법이다.
   4. 알고리즘을 평가하는 기준으로 시간복잡도와 공간복잡도가 있다.

* c

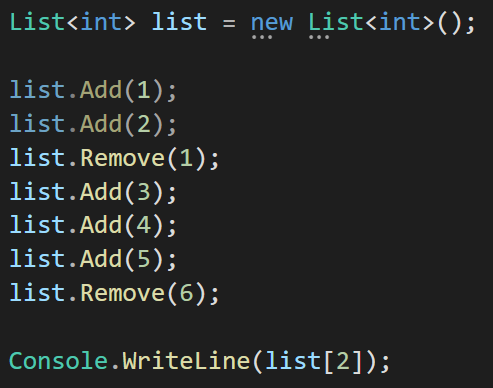
1. 다음 중 자료구조에 대한 설명으로 옳지 않은 것을 고르시오.
   1. 자료구조란 자료를 더 효율적으로 저장하고, 관리하기 위해 사용하며, 저장하는 방법에 따라 필요하는 작업의 수행시간을 단축시켜준다.
   2. 개발자는 상황에 따라 더 빈번하게 일어날 수 있는 기능에 유리한 자료구조를 선정하여 프로그램을 구현해야 한다.
   3. 모든 자료구조는 데이터를 연속적으로 저장하고 관리하기 때문에 자료구조를 만들 때 모든 데이터를 수용할 수 있을만한 큰 공간을 메모리에 연속적으로 확보해야 한다.
   4. 선형 자료구조는 하나의 자료 다음에 하나의 자료가 있는 일직선형 자료구조이며, 비선형 자료구조는 하나의 자료 다음에 둘 이상의 자료가 있는 자료구조이다.

* d

1. 다음 중 리스트에 대한 설명으로 옳지 않은 것을 고르시오.
   1. 리스트는 배열을 이용하여 구현한 자료구조로, 배열이 런타임 중 크기를 변경할 수 없다는 점을 개선한 런타임 중 크기를 확장할 수 있는 자료구조이다.
   2. 리스트는 데이터가 추가되는 경우 필요한 메모리의 공간만큼 힙영역에 할당받았던 메모리의 끝부분을 연장하여 추가 데이터를 보관한다.
   3. 리스트는 데이터수(Count)와 용량(Capacity)를 가지고 있으며, 데이터수가 용량을 넘어서는 경우 리스트는 추가적인 데이터를 보관하기 위한 작업을 진행한다.
   4. 리스트는 배열기반의 자료구조이므로 인덱스 사용이 가능하다.

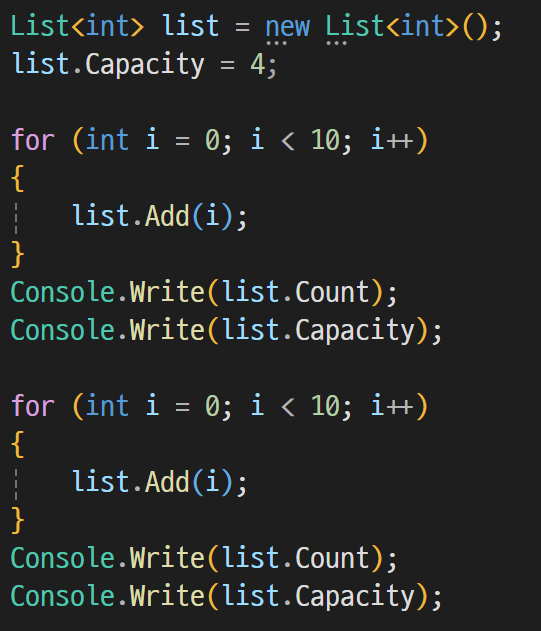
* b

1. 다음 코드의 출력을 적으시오. 4



* 4

1. 다음 코드의 출력을 적으시오. 10 16 20 32

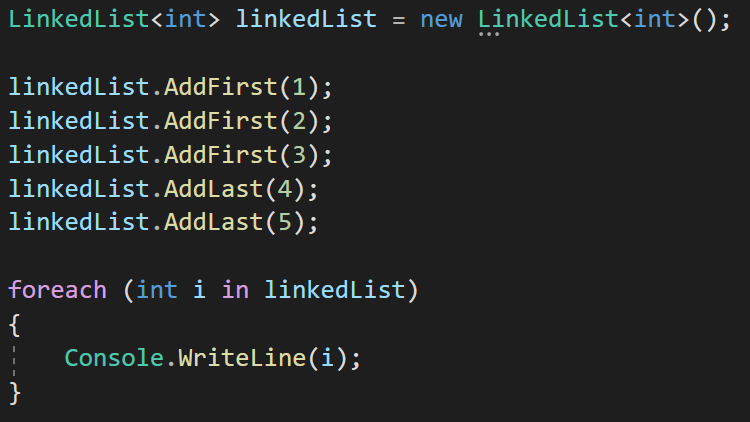


* 10 16 20 32

1. 다음 중 연결리스트에 대한 설명으로 옳지 않은 것을 고르시오. d (b 는 오타이긴 하나 틀린말이 될 수 있음. 노드 추가가 아닌 연결, 연결해제)
   1. 연결리스트는 데이터를 포함하는 노드를 기본 저장단위로 사용하는 자료구조이다.
   2. 연결리스트는 노드를 통해 데이터를 연속적으로 저장하지 않고 삽입&삭제 당시 노드를 추가하여 연결구조만 구성하기에, 삽입&삭제에 시간소요가 적다.
   3. 연결리스트는 노드가 다음 노드만을 참조하는 단방향 연결리스트, 이전&다음 노드를 참조하는 양방향 연결리스트, 처음 노드와 마지막 노드가 서로를 참조하여 순환구조를 가지는 원형 연결리스트가 있다.
   4. 연결리스트는 인덱스 사용이 가능하다.

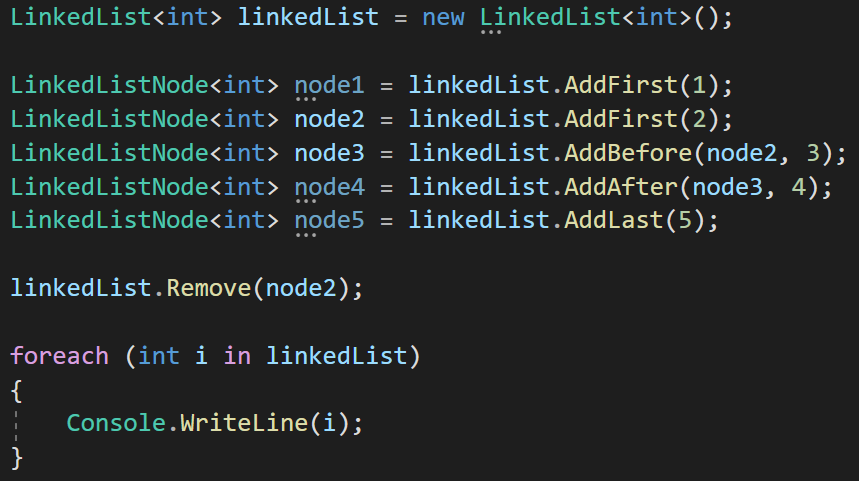
* d

1. 다음 코드의 출력을 적으시오. 3,2,1,4,5



* 3 2 1 4 5

1. 다음 코드의 출력을 적으시오.

 3 4 1 5

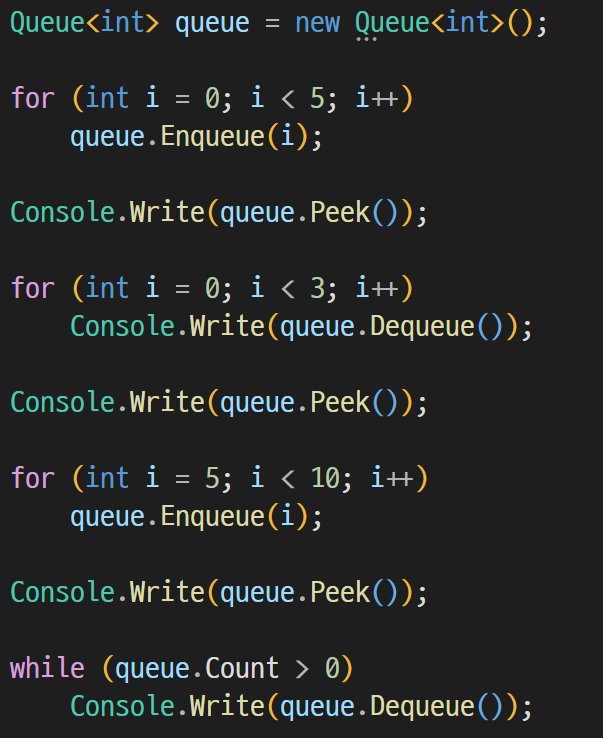
1. 스택과 큐의 데이터 입출력 순서 특성을 적으시오. 또한 그 특성대로 1, 2, 3, 4, 5가 순서대로 추가되었을 때, 데이터가 꺼내지는 순서를 쓰시오.

* 스택 : 5 4 3 2 1 선입후출, 후입선출 (FILO, LIFO) / 5 4 3 2 1
* 큐 : 1 2 3 4 5 선입선출, 후입후출 (FIFO, LILO) / 1, 2, 3, 4, 5

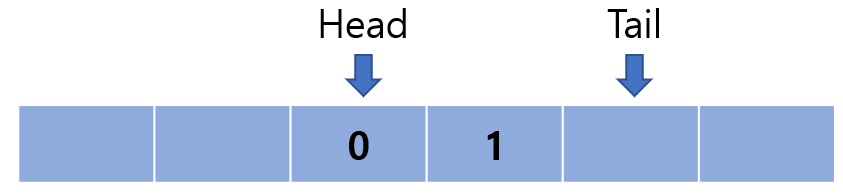
1. 다음 중 스택과 큐에 대한 설명으로 옳지 않은 것을 고르시오. c
   1. 스택은 배열을 이용하여 구현하거나, 이미 구현한 선형리스트의 인터페이스를 스택용 인터페이스로 구성하는 어댑터패턴을 적용하여 구현한다.
   2. 큐는 배열을 이용하여 데이터가 출력될 인덱스를 head, 데이터가 입력될 인덱스를 tail로 표시하고, 입력&출력이 있을 경우 각 인덱스를 한칸씩 이동시키며 관리한다.
   3. 큐는 배열을 이용하여 구현이 하기에, 배열을 크기만큼 데이터의 입출력이 있었던 경우 사용할 공간이 없어지므로 더이상 큐로 사용이 불가해진다.
   4. 큐는 head와 tail이 일치하는 경우 데이터가 없는 경우와 가득차있는 경우를 구분할 수 없기 때문에, tail이 head 직전칸에 있는 경우 가득차있는 경우로 판정한다.

* d

1. 다음 코드의 출력을 적으시오. 0 0 1 2 3 3 3 4 5 6 7 8 9

 0 0 1 2 3 3 3 4 5 6 7 8 9

1. 다음 큐에서 아래의 과정을 단계별로 진행하시오. (크기가 6인 순환배열을 사용하는 상황)  
   표의 첫번째행은 Head와 Tail 위치를 표시하고, 두번째행은 데이터를 표시하시오.  
   Enqueue(2), Enqueue(3), Dequeue(), Dequeue(), Enqueue(4), Enqueue(5), Dequeue()



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Head |  | Tail | Tail |
|  |  | 0 | 1 | 2 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tail |  | Head |  |  | Tail |
|  |  | 0 | 1 | 2 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tail |  |  | Head |  | Tail |
|  |  |  | 1 | 2 | 3 |

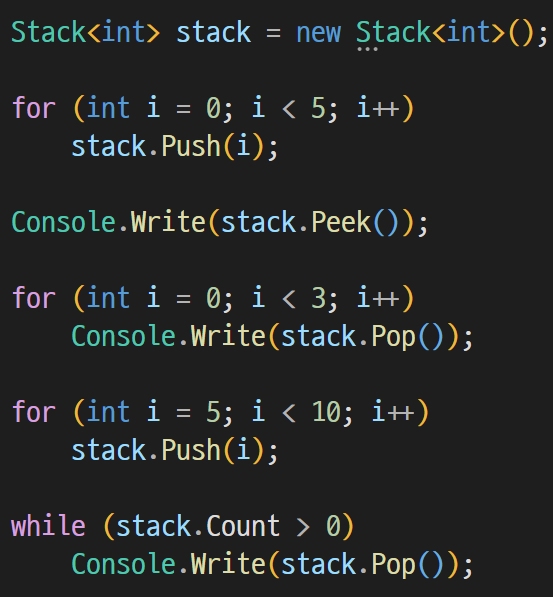
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tail |  |  |  | Head | Tail |
|  |  |  |  | 2 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tail | Tail |  |  | Head |  |
| 4 |  |  |  | 2 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Tail | Tail |  | Head |  |
| 4 | 5 |  |  | 2 | 3 |

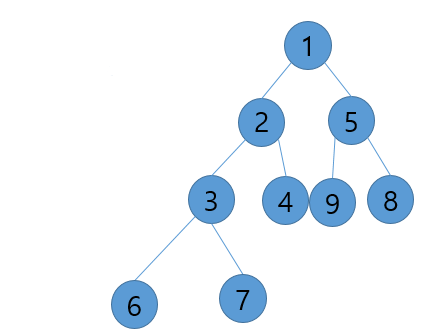
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Tail | Tail |  |  | Head |
| 4 | 5 |  |  |  | 3 |

1. 다음 아래의 코드의 출력을 적으시오.

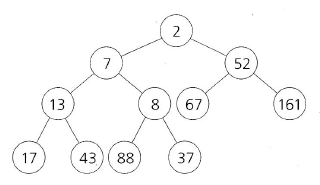


4 4 3 2 5

1. 우선순위큐(힙)에 아래의 순서대로 데이터가 추가될 경우를 그리시오. (우선순위는 값이 작을수록 높다)  
   <3, 1, 8, 6, 4, 9, 5, 2, 7>



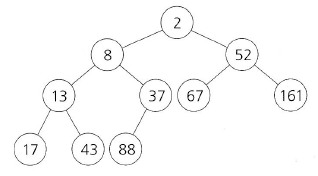
1. 다음 우선순위큐(힙)을 배열로 구현하고, 각 노드의 부모노드, 왼쪽자식노드, 오른쪽자식노드를 참조하기 위한 처리방법을 구하시오.



[2, 7, 52, 13, 8, 67, 161, 17, 43, 88, 37] i = index

* 왼쪽자식노드 : CLi = (i+1)\*2 -1 index \*2 +1
* 오른쪽자식노드 : CRi = (i+1)\*2 index \* 2 +2
* 부모노드 : Pi = (i-1)/2 (index -1)/2

1. 다음 우선순위큐(힙)에서 데이터 7이 추가된 경우를 단계별로 진행하시오.

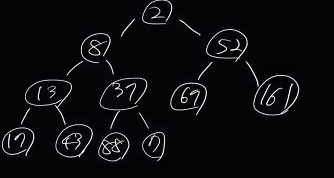
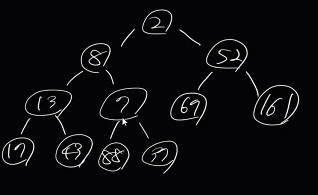
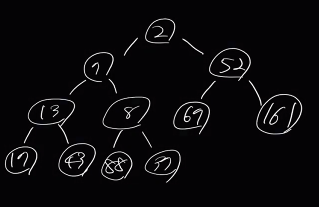


37 오른쪽노드로 7 삽입

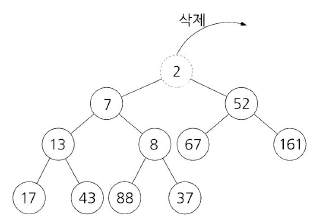
7과 37 비교 -> 7과 37 교환

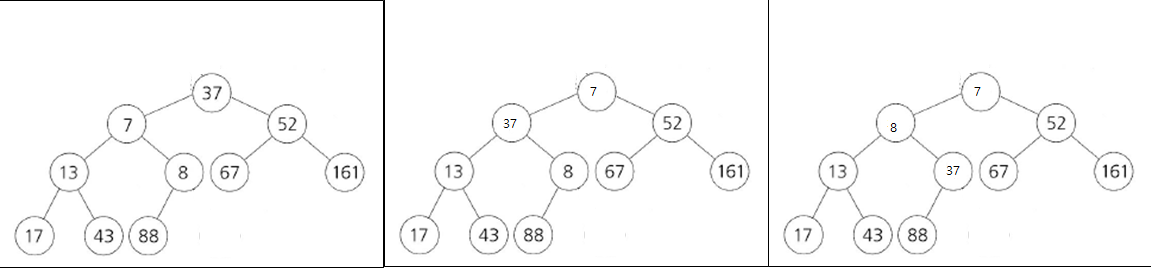
7과 8 비교 -> 7과 8 교환

7과 2 비교 -> 7위치 결정

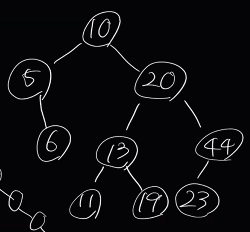
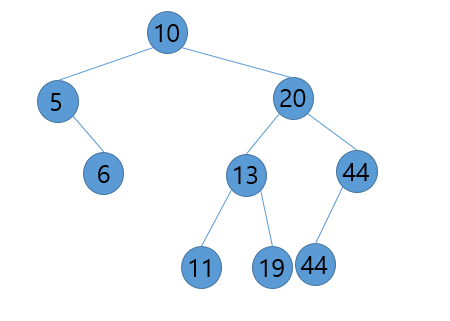


1. 다음 우선순위큐(힙)에서 가장우선순위가 높은 데이터를 제거했을 경우를 단계별로 진행하시오.

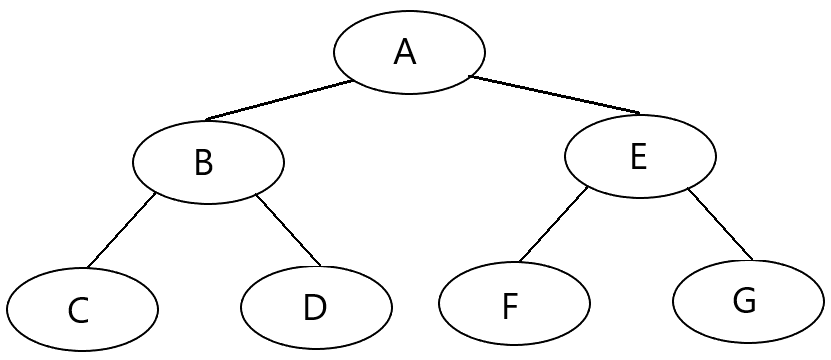




1. 이진탐색트리에 아래의 순서대로 데이터를 추가했을 경우 이진탐색트리의 상태를 그리시오.  
   <10, 5, 20, 13, 11, 6, 19, 44, 23>



1. 다음 이진탐색트리의 선위순회, 중위순회, 후위순회시 출력을 적으시오.



* 선위순회 : c d b a f e g 노드 왼쪽 오른쪽 A B C D E F G
* 중위순회 : a b c d e f g 왼쪽 노드 오른쪽 C B D A F E G
* 후위순회 : c d b f g e a 왼쪽 오른쪽 노드 C D B F G E A

1. 다음 중 해시테이블에 대한 설명으로 옳지 않은 것을 고르시오. D
   1. 해시테이블은 키 값을 해시함수로 해싱하여 해시테이블의 특정 위치로 직접 접근하는 방식이다.
   2. 해시는 임의의 길이를 가진 데이터를 고정된 길이를 가진 데이터로 변환하는 과정을 의미한다.
   3. 해시함수의 처리속도가 빠를수록 효율이 좋으며, 밀집도가 낮을수록 충돌이 적게 발생하므로 효율적이다.
   4. 해시함수는 모든 입력에 대응하는 출력을 1대1로 가져야 하며, 항상 동일한 값을 반환해야 한다.

* d

1. 다음 중 해시테이블의 충돌에 대한 설명으로 옳지 않은 것을 고르시오. D 데이터를 그 위치 그대로 새로운 배열에 복사해준다. (X) -> 그 위치가 아닌 전부 새로운 배열로 재해싱을 한다.
   1. 해시함수가 서로 다른 입력값에 대해 동일한 해시테이블 주소를 반환하는 것을 충돌이라 한다.
   2. 체이닝은 충돌이 발생하는 경우 연결리스트로 데이터들을 연결하는 방식이며, 해시테이블에 자료가 많아지더라도 성능저하가 적다는 장점이 있지만, 추가적인 저장공간이 필요하다는 단점이 있다.
   3. 개방주소법은 충돌이 발생하는 경우 다른 빈공간에 데이터를 삽입하는 방식이며, 추가적인 저장공간이 필요하지 않다는 장점이 있지만, 해시테이블에 자료가 많아질수록 성능저하가 많다는 단점이 있다.
   4. 개방주소법의 경우 공간 사용률이 높을 경우 성능저하가 발생하므로, 일정비율 이상 데이터가 추가되는 경우 해시테이블의 크기를 늘리고 테이블 내의 모든 데이터를 그 위치 그대로 새로운 배열에 복사해준다.

* b

1. 다음 해시테이블에 아래의 순서대로 데이터가 추가될 경우, 해시테이블의 데이터 저장상태를 구하시오.  
   (해시테이블의 크기는 20, 해시함수는 나눗셈법을 사용하며, 충돌해결방안은 열린주소(선형탐사)로 진행한다.)  
   Add(12, 사과), Add(87, 배), Add(53, 감), Add(32, 딸기)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
|  | 1, 감 |  |  |  |  |  | 87, 배 |  |  |
| [10] | [11] | [12] | [13] | [14] | [15] | [16] | [17] | [18] | [19] |
|  |  | 12,사과  32, 딸기 | 93, 배, 53 감 | 53 감 | 32, 딸기 |  |  |  |  |

1. 다음 아래의 순서로 배열에 저장되어 있을 경우 선택정렬을 단계별로 진행하시오.  
   (단계마다 선택되는 배열요소를 표시하고, 스크린샷 없이 변경되는 과정을 텍스트로 적을 것)  
   <5, 3, 7, 1, 9, 2, 8, 4, 6>

<1, 5, 3, 7, 9, 2, 8, 4, 6>

<1, 2, 5, 3, 7, 9, 8, 4, 6>

<1, 2, 3, 5, 7, 9, 8, 4, 6>

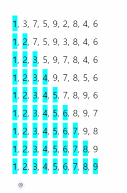
<1, 2, 3, 4, 5, 7, 9, 8, 6>

<1, 2, 3, 4, 5, 7, 9, 8, 6>

<1, 2, 3, 4, 5, 6, 7, 9, 8>

<1, 2, 3, 4, 5, 6, 7, 8, 9>

<1, 2, 3, 4, 5, 6, 7, 8, 9>



1. 다음 아래의 순서로 배열에 저장되어 있을 경우 삽입정렬을 단계별로 진행하시오.  
   (단계마다 삽입되는 요소를 표현하고, 스크린샷 없이 변경되는 과정을 텍스트로 적을 것)  
   <5, 3, 7, 1, 9, 2, 8, 4, 6>

<3, 5, 7, 1, 9, 2, 8, 4, 6>

<1, 3, 5, 7, 9, 2, 8, 4, 6>

<1, 2, 3, 5, 7, 9, 8, 4, 6>

<1, 2, 3, 5, 7, 8, 9, 4, 6>

<1, 2, 3, 4, 5, 7, 8, 9, 6>

<1, 2, 3, 4, 5, 6, 7, 8, 9>

1. 다음 아래의 순서로 배열에 저장되어 있을 경우 버블정렬을 단계별로 진행하시오.  
   (단계마다 가장 우측에 배치되는 요소를 표현하고, 스크린샷 없이 변경되는 과정을 텍스트로 적을 것)  
   <5, 3, 7, 1, 9, 2, 8, 4, 6>

<3, 5, 7, 1, 9, 2, 8, 4, 6>

<3, 5, 1, 7, 9, 2, 8, 4, 6>

<3, 5, 1, 7, 2, 9, 8, 4, 6>

<3, 5, 1, 7, 2, 8, 9, 4, 6>

<3, 5, 1, 7, 2, 8, 4, 9, 6>

<3, 5, 1, 7, 2, 8, 4, 6, 9>

<3, 1, 5, 7, 2, 8, 4, 6, 9>

<3, 1, 5, 2, 7, 8, 4, 6, 9>

<3, 1, 5, 2, 7, 4, 8, 6, 9>

<3, 1, 5, 2, 7, 4, 6, 8, 9>

<1, 3, 5, 2, 7, 4, 6, 8, 9>

<1, 3, 2, 5, 7, 4, 6, 8, 9>

<1, 3, 2, 5, 4, 7, 6, 8, 9>

<1, 3, 2, 5, 4, 6, 7, 8, 9>

<1, 2, 3, 5, 4, 6, 7, 8, 9>

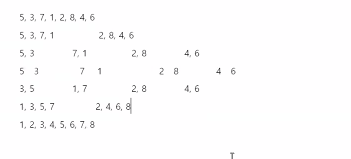
<1, 2, 3, 4, 5, 6, 7, 8, 9>

1. 다음 아래의 순서로 배열에 저장되어 있을 경우 병합정렬을 단계별로 진행하시오.  
   (각 단계 중 배열이 나뉜 것을 표현하고, 스크린샷 없이 변경되는 과정을 텍스트로 적을 것)  
   <5, 3, 7, 1, 2, 8, 4, 6>

<3, 5, 1, 7, 2, 8, 4, 6>

<1, 3, 5, 7, 2, 4, 6, 8>

<1, 2, 3, 4, 5, 6, 7, 8>



임시 배열을 만들기에 공간 복잡도 n인 것이 단점

1. 다음 아래의 순서로 배열에 저장되어 있을 경우 퀵정렬을 단계별로 진행하시오.  
   (각 단계 중 pivot을 표시하고, 스크린샷 없이 변경되는 과정을 텍스트로 적을 것)  
   <”5’” 3, 7, 1, 9, 2, 8, 4, 6>

<5, 3, 1, 4, 2, 9, 8, 6, 7>

<3, 1, 4, 2, 5, 9, 8, 6, 7>

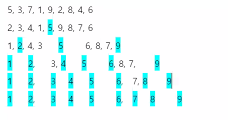
<”3”, 1, 2, 4, 5, 9, 8, 6, 7>

< 1, 2, 3, 4, 5, 9, 8, 6, 7>

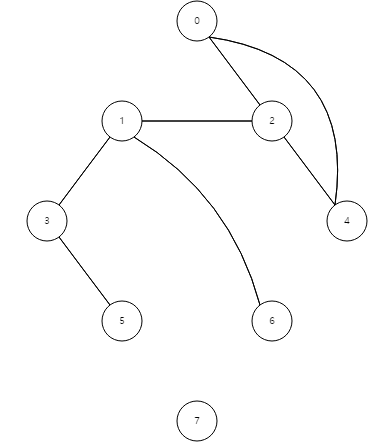
< 1, 2, 3, 4, 5, 9, 8, 6, “7”>

< 1, 2, 3, 4, 5, 6, 8, 9, “7”>

< 1, 2, 3, 4, 5, 6, 7, 8, 9>



1. 아래와 같이 그래프가 있을 경우, 이를 인접행렬 그래프로 표현하시오.



{ false, false, true, false, true, false, false, false},

{ false, false, true, true, false, false, true, false},

{ false, false, false, false, false, false, false, false},

{ false, true, false, false, false, true, false, false},

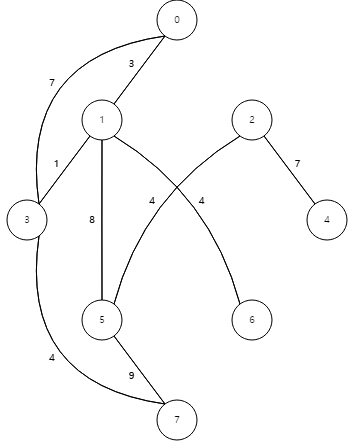
{ true, false, true, false, false, false, false, false},

{ false, false, false, true, false, false, false, false},

{ false, true, false, false, false, false, false, false},

{ false, false, false, false, false, false, false, false},

1. 아래와 같이 가중치가 있는 그래프가 있을 경우, 이를 인접행렬 그래프로 표현하시오.



{ 0, 3, INF, 7, INF, INF, INF, INF},

{ 3, 0, INF, 1, INF, 8, 4, INF},

{ INF, INF, 0, INF, 7, 4, INF, INF},

{ 7, 1, INF, 0, INF, INF, INF, 4},

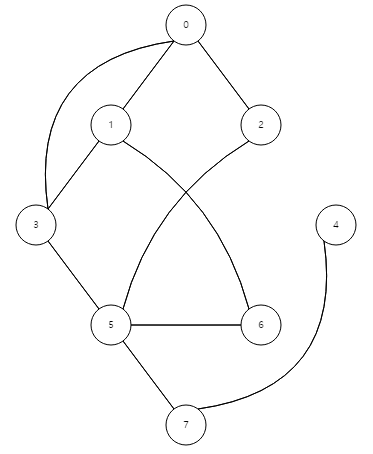
{ INF, INF, 7, INF, 0, INF, INF, INF},

{ INF, 8, 4, INF, INF, 0, INF, 9},

{ INF, 4, INF, INF, INF, INF, 0, INF},

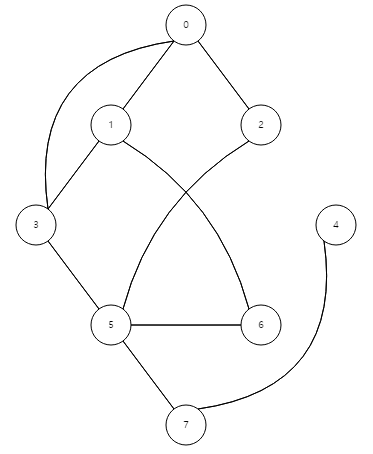
{ INF, INF, INF, 4, INF, 9, INF, 0},

1. 아래 그래프를 깊이우선탐색할 경우, 정점의 탐색순서를 구하시오. 0 1 3 5 2 6 7 4



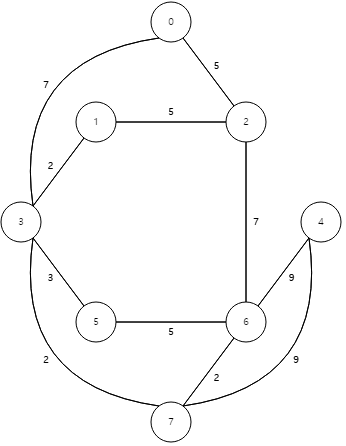
0 1 3 5 6 7 4

1. 아래 그래프를 너비우선탐색할 경우, 정점의 탐색순서를 구하시오.



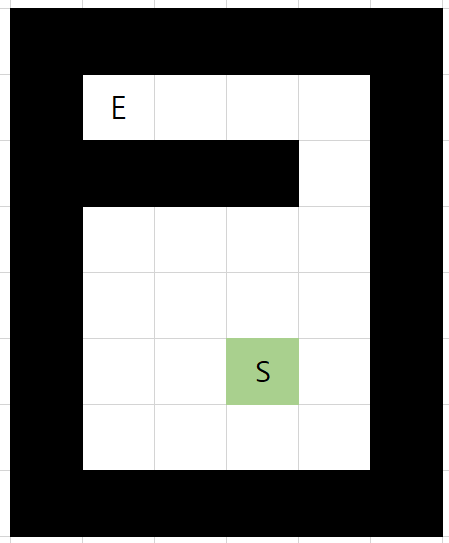
0 1 2 3 6 5 7 4

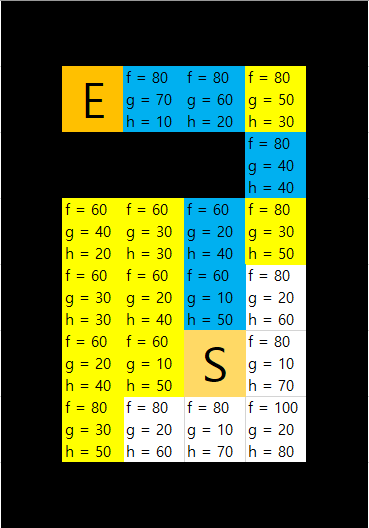
1. 아래 그래프를 0 정점부터 다익스트라 알고리즘으로 최단거리를 구할 경우 아래의 배열을 채우시오.



|  |  |  |  |
| --- | --- | --- | --- |
| vertex | bool[] visited | int[] distance | int[] parents |
| 0 | true | 0 | -1 |
| 1 | true | 7 2 | 0 3 |
| 2 | true | 5 | 0 |
| 3 | true | 7 | 0 |
| 4 | true | 7 2 9 | 0 3 7 4 |
| 5 | true | 7 3 | 0 3 |
| 6 | true | 5 7 | 0 2 |
| 7 | true | 7 2 | 0 3 |

1. 아래의 타일맵을 S 지점부터 E 지점까지 A\*알고리즘을 통해 최단경로를 구하시오. 각 정점들 중 탐색한 정점에 의해 부여되는 f, g, h 값을 빈칸에 넣고, 탐색여부(파랑), 최단경로(노랑) 표시하시오.  
   단, 대각선 이동을 인접 벽의 여부와 상관없이 모두 허용하는 8방향 이동을 진행하며, 휴리스틱은 맨허튼 거리를 기준으로 측정한다.





휴리스틱이 맨해튼 -> 휴리스틱은 계산할 경우 맨해튼 방식을 사용하고 g는 실제값을 주는 방식으로 계산한다.

