Data Structure AVL Heaps Implementation

Header

|  |
| --- |
| Heap  이중 포인터. 배열에 접근하면 또 다른 데이터를 가리키는 포인터가 있기 때문.  Last -> 포인터 배열 중 데이터를 가리키는 포인터 역할  Size -> 포인터 배열 중 데이터가 저장된 할당량 갯수.  Compare -> 포인터들을 비교하는 함수  maxSize -> 포인터 배열의 맥시멈 크기 |

Create Heap

|  |
| --- |
| Heap을 이용하기 위한 초기화   1. 동적할당을 통해 heap 메모리 할당. 2. 할당된 heap 각 데이터 초기화.   포인터배열[0]이 시작점이기 때문에 초기화할 때는 last는 -1로 초기화.   1. Maxsize, heapAry 크기 초기화 |

Insert

|  |
| --- |
| 힙에 데이터를 넣는 작업.  데이터를 삽입한 후 reheapup함수를 불러 재정렬함. |

reheapup

|  |
| --- |
| 부모 노드와 자식노드를 비교하여 자식노드가 클 경우 부모 노드와 자리를 변경한다. 이후 부모노드를 대상으로 함수를 재귀적 적용하여 heap 전체가 정렬되게 만듬. |

heapDelete

|  |
| --- |
| 힙의 루트를 제거하고 그 저장된 데이터 값을 반납함. 이후 heapAry 마지막에 저장된  데이터를 다시 루트로 삼고 reheapdown함수를 적용하여 정렬함. |

reheapDown

|  |
| --- |
| reheapUp과 반대로 적용하는 함수.  부모노드와 자식노드에 저장된 데이터 값을 비교하여 자식노드에 해당하는 값이 작을 경우 부모노드와 자식노드의 위치를 바꾼다. 이후 다시 reheapDown을 재귀적 적용. |

HeapDestroy

|  |
| --- |
| HeapAry와 heap을 대상으로 free함수 적용 |

heapPrint

|  |
| --- |
| 할당된 포인터 last 크기만큼 각 데이터를 출력. |

heapEmpty

|  |
| --- |
| Heap에 데이터가 없을 경우, 참을 반납, 아닐 경우 거짓 반납 |

HeapFull

|  |
| --- |
| 힙에 더이상 데이터가 들어갈 수 없을 경우, 즉 size 크기와 maxsize가 같을 경우 참 반납, 아닐 경우 거짓 반납 |

HeapCount

|  |
| --- |
| 힙에 할당된 데이터 갯수를 출력. |