자료구조론

Merge Sort / Heap Sort

u@hataeseong-ui-MacBook-Pro:~/Desktop/2017_CSE2010_2016025041/HW9$./a.out
merge_sort:
1 5 11 15 19 26 48 59 61 77
u@hataeseong-ui-MacBook-Pro:~/Desktop/2017_CSE2010_2016025041/HW9$./a.out
heap_sort:
1 7 21 45 51 53 73 109 410 578
- 실행 결과 일치

```
void merge(int list[], int left, int mid, int right)
{
  int i, j, k, l;
  i=left;  j=mid+1;  k=left;

  int tmp[MAX_SIZE];
  for(; i <= mid && j <= right; k++){
   if(list[i] <= list[j]) //두 리스트의 값을 비교한 뒤 작은 값 삽입
    tmp[k] = list[i++];
   else
    tmp[k] = list[j++];
  }

  while(i < mid + 1) //한 리스트가 비면 남은 리스트의 값 삽입
   tmp[k++] = list[i++];
  while(j <= right)
   tmp[k++] = list[j++];

 //  memmove(&list[left], &tmp[left], sizeof(int) * (right - left));
 for(int i = left; i < k; i++){
  list[i] = tmp[i];
 }
}
```

- 두 리스트의 인덱스 0부터 비교해가면서 작은 값부터 차례대로 넣은뒤 한 리스트의 끝까지가면 나머지 남은 리스트에 있는 값을 넣고 지역변수로 선언한 리스트에 넣었기때문에 값을 복사 / 혹은 메모리를 복사해서 기존 포인터 list[]에 넣는다.

```
void merge_sort(int list[], int left, int right)
{
  int mid;
  if(left<right){
    mid = (left+right)/2;     /* 리스트의 균등 분할 */
    merge_sort(list, left, mid);    /* 부분 리스트 정렬 */
    merge_sort(list, mid+1, right); /* 부분 리스트 정렬 */
    merge(list, left, mid, right);    /* 합병 */
  }
}
```

- merge_sort를 재귀형식으로 짜 배열을 계속 반으로 나누어 리스트당 하나의 값을 가질때까지 나눈 뒤 위에서 짠 함수를 통해 합병하면서 정렬한다.

```
void adjust(int heap[], int root, int n)
{
 int child, temp;
  temp = heap[root];   // 루트값저장
  child = 2 * root;      // 왼쪽 자식노드
  while(child <= n) {  // 마지막 노드까지 반복
    if(child < n && heap[child] < heap[child+1])   // 더 작은 자식 노드
      child++;
    if(temp>heap[child]) break;   // 부모노드와 자식노드 비교
    else  // 자식노드값을 부모노드로 복사
      heap[child / 2] = heap[child];
    child = child * 2;  // 한 레벨 아래로 이동
  }
  heap[child / 2] = temp;
}
```

- 가장 큰 값을 찾아 부모 노드로 옮기는, 최대 힙 트리를 만드는 작업

```
void heap_sort(int list[], int n)
{
  int i, temp;
  int heap[MAX_SIZE + 1];

  for(i=0;i<n;i++)
    heap[i+1]=list[i];
```

```
    for(i=n/2;i>0;i--)    // 주어진 리스트를 최대힙으로 변환
      adjust(heap,i,n);
    for(i=n-1;i>0;i--) {    // 루트노드와 마지막 노드 교환
      SWAP(heap[1], heap[i+1], temp);
      adjust(heap, 1, i);    // 축소된 리스트를 루트노드부터 재조정
    }
    for(i=0;i<n;i++)
      list[i]=heap[i+1];
  }
```

- 리스트를 최대힙으로 만든 뒤 루트를 빼내고 다시 최대힙으로 만들고 루트를 빼내는 작업을 반복, 제일 큰 값(루트)을 마지막 노드로 옮기는 작업을 반복해 끝까지하게되면 오름차순 정렬 끝