자료구조 (Data Structure)
6주차: 힙

지난 시간 요약

- 최대힙: 자식이 부모보다 데이터가 작은 이진 트리
- 최소힙: 자식이 부모보다 데이터가 큰 이진 트리

최대힙 데이터 추가

1) 배열의 마지막에 새 데이터를 추가한다

2) 새 데이터가 부모보다 크면 교환한다

3) 새 데이터가 루트가 되거나, 부모보다 작거나 같을 때까지 반복한다

최대힙 데이터 삭제

1) 루트 데이터를 반환한다

2) 배열의 마지막 데이터를 루트에 복사한다

3) 복사된 데이터가 자식보다 작으면 교환한다

4) 복사된 데이터가 트리 끝에 도달하거나, 자식보다 크거나 같을 때까지 3)을 반복한다

최소힙 데이터 추가

1) 배열의 마지막에 새 데이터를 추가한다

2) 새 데이터를 부모와 비교하여, 부모가 더 크면 교환한다

3) 새 데이터가 루트가 되거나, 부모가 더 작거나 같을 때까지 2)를 반복한다

최소힙 데이터 삭제

- 1) 루트 데이터를 반환한다
- 2) 배열의 마지막 데이터를 루트에 복사한다
- 3) 복사한 데이터를 자식과 비교하여, 자식이 더 작으면 교환한다
- 4) 복사한 데이터가 끝에 도달하거나, 자식이 더 크거나 같을 때까지 3)을 반복한다

오늘 목표

1) 실무에서 구현된 힙 (리눅스 소스 코드)

2) 중간고사 문제 풀이

실무에서 구현된 힙

- 출처: github.com/torvalds/linux

- 파일: include/linux/min_heap.h
- 버전: 최초

- Min Heap 코드는 최소 힙을 관리하는 기능을 제공
- 최소 힙은 이진 트리 구조로, 각 노드의 값이 자식 노드들의 값보다 작거나 같은 특징이 있다.
- 결과적으로 가장 작은 요소가 루트에 있다.

실무에서 구현된 힙 - 구조체

```
struct min_heap {
        int nr;         // 현재 힙에 저장된 데이터 개수
        int size;       // 배열에 저장 가능한 최대 개수
        int *data;      // 데이터를 저장하는 배열의 포인터
};
```

실무에서 구현된 힙 - 초기화

```
void min_heap_init(struct min_heap *heap,
                   int *data, int size)
{
        heap->nr = 0;
        heap->size = size;
        heap->data = data;
}
```

실무에서 구현된 힙 - 초기화 테스트

- 다음 코드를 실행한 후 heap의 상태는?

```
int values[] = {3, 1, 2, 4};
struct min_heap heap = {
        .nr = 0,
        .size = sizeof(values) / sizeof(values[0]),
        .data = values,
};
```

실무에서 구현된 힙 - 기능 확장 방법

```
struct min_heap_callbacks {
        /* 데이터 비교 함수, 거꾸로 하면 최대힙 */
        int (*less)(const int *lhs, const int *rhs);

        /* 데이터 교환 함수, 구조체 데이터에 필수 */
        void (*swap)(int *lhs, int *rhs);
};
```

실무에서 구현된 힙 - 작다 함수, 크다 함수

```
int less_than(const int *lhs, const int *rhs)
{
        return *lhs < *rhs;
}

int greater_than(const int *lhs, const int *rhs)
{
        return *lhs > *rhs;
}
```

실무에서 구현된 힙 - 작다 함수, 크다 함수

- 다음 코드를 실행한 후 x와 y의 값은?

```
int numbers[] = {10, 20, 30};
int x = less_than(&numbers[0], &numbers[1]);
int y = greater_than(&numbers[0], &numbers[1]);
```

실무에서 구현된 힙 - 교환 함수

```
void swap_int(int *lhs, int *rhs)
{
        int temp = *lhs;
        *lhs = *rhs;
        *rhs = temp;
}
```

실무에서 구현된 힙 - 교환 함수

- 다음 코드를 실행한 후 numbers 배열의 상태는?

```
int numbers[] = {10, 20, 30};
swap_int(&numbers[0], &numbers[2]);
```

실무에서 구현된 힙 - 힙으로 만들기

```
void min_heapify_all(struct min_heap *heap,
                const struct min_heap_callbacks *funcs)
{
        int i;

        for (i = heap->nr / 2 - 1; i >= 0; i--)
                min_heap_sift_down(heap, i, funcs);
}
```

실무에서 구현된 힙 - 데이터 내리기 1 / 4

```
void min_heap_sift_down(struct min_heap *heap,
                        int pos,
                        const struct min_heap_callbacks *funcs)
{
        int *left, *right, *parent, *smallest;
        int *data = heap->data;
        for (;;) {
                if (pos * 2 + 1 >= heap->nr)
                        break;
```

실무에서 구현된 힙 - 데이터 내리기 2 / 4

```
left = data + (pos * 2 + 1);
parent = data + pos;
smallest = parent;

if (funcs->less(left, smallest))
        smallest = left;
```

실무에서 구현된 힙 - 데이터 내리기 3 / 4

```
if (pos * 2 + 2 < heap->nr) {
        right = data + (pos * 2 + 2);
        if (funcs->less(right, smallest))
                smallest = right;
}

if (smallest == parent)
        break;
```

실무에서 구현된 힙 - 데이터 내리기 4 / 4

```
                funcs->swap(smallest, parent);
                if (smallest == left)
                        pos = pos * 2 + 1;
                else
                        pos = pos * 2 + 2;
        }
}
```

실무에서 구현된 힙 - 데이터 내리기

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
min_heap_sift_down(&heap, 3, &callbacks);
```

실무에서 구현된 힙 - 데이터 내리기

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
min_heap_sift_down(&heap, 2, &callbacks);
```

실무에서 구현된 힙 - 데이터 내리기

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
min_heap_sift_down(&heap, 1, &callbacks);
```

실무에서 구현된 힙 - 데이터 내리기

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
min_heap_sift_down(&heap, 0, &callbacks);
```

실무에서 구현된 힙 - 힙으로 만들기

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
min_heapify_all(&heap, &callbacks);
```

실무에서 구현된 힙 - 데이터 삭제 1 / 2

```
void min_heap_pop(struct min_heap *heap,
                  const struct min_heap_callbacks *funcs)
{
        int *data = heap->data;
        if (heap->nr <= 0) {
                printf("Popping an empty heap\n");
                return;
        }
```

실무에서 구현된 힙 - 데이터 삭제 2 / 2

```
        heap->nr--;
        data[0] = data[heap->nr];
        min_heap_sift_down(heap, 0, funcs);
}
```

실무에서 구현된 힙 - 데이터 삭제

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
min_heapify_all(&heap, &callbacks);
min_heap_pop(&heap, &callbacks);
min_heap_pop(&heap, &callbacks);
min_heap_pop(&heap, &callbacks);
```

실무에서 구현된 힙 - 데이터 추가 1 / 2

```
void min_heap_push(struct min_heap *heap, int val,
                   const struct min_heap_callbacks *funcs)
{
        int *data = heap->data;
        int *child, *parent;
        int pos;
        if (heap->nr >= heap->size) {
                printf("Pushing into a full heap\n");
                return;
        }
```

실무에서 구현된 힙 - 데이터 추가 2 / 2

```
        pos = heap->nr;
        data[pos] = val;
        heap->nr++;
        for (; pos > 0; pos = (pos - 1) / 2) {
                child = data + pos;
                parent = data + (pos - 1) / 2;
                if (funcs->less(parent, child)) break;
                funcs->swap(parent, child);
        }
}
```

실무에서 구현된 힙 - 데이터 추가

- 다음 코드를 실행한 후 heap의 상태는?

```
int data[] = {3, 1, 2, 4};
struct min_heap heap = {.nr = 0, .size = 4, .data = data};
struct min_heap_callbacks fs = {
        .less = less_than, .swap = swap_int, };
heap.nr = 4;
int values[] = {3, 4, 1, 2};
for (int i = 0; i < 4; i++) {
        min_heap_push(&heap, values[i], &callbacks);
}
```

2018년 중간고사 문제 1. Time complexity

• 다음에 주어진 code들에 대해 time complexity를 big-oh (O) notation을 사용하여 표현하시오.
그리고, 그렇게 판단하는 이유도 함께 기술하시오.
(주: 각 3점씩, 수식이 맞아도 이유가 잘못되면 0점)

2018년 중간고사 문제 1-(1)

```
int f1(int n)
{
        int result = n * n;
        return result;
}
```

2018년 중간고사 문제 1-(2)

```
int f2(int n)
{
        int result = 0;
        while (n > 1) {
                n = n / 2;
                result += 1;
        }
        return result;
}
```

2018년 중간고사 문제 1-(3)

```
int f3(int n, int m)
{
        int result = 0;
        int i, j;
        for (i = 1; i < n; i *= 2)
                result += 1;
        for (j = 1; j < m; j *= 5)
                result += 1;
        return result;
}
```

2018년 중간고사 문제 2. Circular Queue

- 크기가 6인 circular queue에 다음과 같이
삽입과 삭제가 순서대로 진행되었을 경우,
각 단계에서의 circular queue의 내용을 적으시오.
(각 2점씩) 단, front와 rear는 초기값으로 0을 부여한다.

- enqueue(1), enqueue(2), enqueue(3)
- enqueue(7), enqueue(6), enqueue(5)
- dequeue(), dequeue()
- enqueue(4)

2018년 중간고사 문제 3. Stack

- 다음과 같이 infix notation을 postfix notation으로 변경할 수 있다.

3+4*2-7 → 342*+7- 2*(4+6)-9 → 246+*9-

- 이러한 기능을 갖는 function을 개발한다고 할 때, 그 function의 알고리즘을 기술하시오.

Flow chart의 형태로 표현해도 좋고, pseudo code, 혹은 논리적으로 명확하고 간단한 우리글도 좋다.
단, 알고리즘대로 절차를 수행했을 때 변환이 완료될 수 있어야 한다.

2018년 중간고사 문제 4. Linked list

- 아래 프로그램에서 ①, ②, ③을 수행하고 난 후 head가 가리키는 linked list가 어떤 모양인지 그림으로 나타내시오.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node* nodePointer;
```

2018년 중간고사 문제 4. Linked list

```
typedef struct node {
        int data;
        nodePointer prev;
        nodePointer next;
} node;
```

2018년 중간고사 문제 4. Linked list

```
nodePointer GetNewNode(int x) {
        nodePointer newNode = malloc(sizeof(node));
        newNode->data = x;
        newNode->prev = NULL;
        newNode->next = NULL;
        return newNode;
}
```

2018년 중간고사 문제 4. Linked list

```
void f1(int x, nodePointer *ptr)
{
        nodePointer newNode = GetNewNode(x);
        if (*ptr == NULL) {
                *ptr = newNode;
                return;
        }
```

2018년 중간고사 문제 4. Linked list

```
        *ptr->prev = newNode;
        newNode->next = *ptr;
        *ptr = newNode;
}
```

2018년 중간고사 문제 4. Linked list

```
void f2(int x, nodePointer *ptr)
{
        nodePointer temp = *ptr;
        nodePointer newNode = GetNewNode(x);
        if (*ptr == NULL) {
                *ptr = newNode;
                return;
        }
```

2018년 중간고사 문제 4. Linked list

```
        while (temp->next != NULL)
                temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
}
```

2018년 중간고사 문제 4. Linked list

```
int main()
{
        nodePointer head = NULL;
        f1(2, &head);
        f2(3, &head);
        f1(4, &head);
        return 0;
}
```

2018년 중간고사 문제 5. Binary tree (12점)

1) Node의 개수가 n일 때 최소 height와 최대 height를 갖는 경우, 그 값이 얼마인지 각각 예를 들어 설명하시오. (6점)

2018년 중간고사 문제 5. Binary tree (12점)

2) Node의 개수가 n일 때 최소 width와 최대 width를 갖는 경우, 그 값이 얼마인지 각각 예를 들어 설명하시오. (6점)

중간고사 문제 6. Binary tree traversal (9점)

- postorder traversal 결과가 D B F G E H C A 이고 inorder traversal 결과가 B D A F E G C H인 binary tree를 그림으로 나타내시오.
  단, 중간 과정을 모두 적으시오.

2018년 중간고사 문제 보너스. (10점)

- 두 자연수의 나누기를 함수로 구현하는데, 빼기를 활용하면 아래 code와 같이 recursion으로 구현하는 것이 가능하다.

연산 결과는 C에서의 '/'의 결과와 같다. 즉, divide(5, 2)의 결과값은 2이고, divide(6, 3)의 결과값은 2이다.

다음 함수의 ①과 ②, ③에 들어갈 한 줄짜리 code를 적으시오.

2018년 중간고사 문제 보너스. (10점)

```
unsigned int divide(unsigned int x, unsigned int y)
{
        if (x < y) /* ① */
        else if (x == y) /* ② */
        else if (x > y) /* ③ */
}
```