VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Cấu Trúc Dữ Liệu và Giải Thuật (DSA)

DSA3 - HK242

Cuối Kì

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 4/2025

Mục lục

1 Lý thuyết cơ bản Heap

Định nghĩa và tính chất

- **Heap** là một cây nhị phân *gần hoàn chỉnh*, tức là:
  - Mọi tầng đều được điền đầy từ trái qua phải, trừ tầng cuối cùng.
  - Tầng cuối cùng có thể chưa đầy nhưng các node vẫn được xếp từ trái sang phải.
- **Phân loại Heap:**
  - **Min-Heap**: Mỗi node cha có giá trị nhỏ hơn hoặc bằng node con.
  - **Max-Heap**: Mỗi node cha có giá trị lớn hơn hoặc bằng node con.

Các công thức quan trọng

- **Chiều cao của Heap** (với n phần tử):

$$h = \lfloor \log_2 n \rfloor + 1$$

- **Số phần tử trong Heap có chiều cao h:**

$$2^{h-1} \leq n \leq 2^h - 1$$

- Nếu Heap là **n-ary heap** (mỗi node có tối đa n con), các công thức trên sẽ thay đổi tương ứng.

Độ phức tạp các thao tác trên Heap

- **Heapify:** $O(\log n)$
- **Build-Heap:** $O(n)$
- **Insert và Remove:** $O(\log n)$
- **Heap Sort:** $O(n \log n)$ (trong cả 3 trường hợp: tốt, xấu, trung bình)

Cách làm bài toán Insert và Remove hiệu quả

- **Nên vẽ cây thay vì vẽ mảng**, vì hình ảnh trực quan sẽ giúp thao tác nhanh hơn.
- **Cách vẽ cây từ một mảng Heap:**
  1. Sắp xếp các phần tử theo từng tầng, từ trái qua phải.
  2. Tầng đầu tiên chứa root, tầng tiếp theo chứa 2 node con của root, tầng kế tiếp chứa 4 node con của 2 node trên, v.v.
- **Cách thực hiện thao tác Insert và Remove:**
  1. **Insert (Thêm phần tử vào Heap)**
     - Thêm vào vị trí cuối cùng của cây (node lá ngoài cùng bên phải).
     - Thực hiện **Reheap Up** (đẩy giá trị lên trên nếu vi phạm tính chất Heap).
  2. **Remove (Xóa phần tử gốc - root)**
     - Swap phần tử root với phần tử cuối cùng.
     - Xóa phần tử cuối cùng (vốn là phần tử root ban đầu).
     - Thực hiện **Reheap Down** (đẩy giá trị xuống dưới nếu vi phạm tính chất Heap).

Lưu ý khi làm bài tập code

- Cần nắm vững các thuật toán **Reheap Up, Reheap Down, Build Heap, và Heap Sort**.
- Thực hành vẽ cây trước khi viết code để dễ hình dung các bước thao tác.

2 Câu hỏi trong các đề thi

1. Biết rằng mảng A dùng để biểu diễn một maxheap và một chỉ số i. Biết rằng việc truy cập các phần tử trong mảng đều hợp lệ, biểu thức nào sau đây LUÔN ĐÚNG?

   a) A[i] > A[2i + 1] b) A[i] < A[2i + 2] c) A[i] > A[2i] d) A[i] < A[2i – 1]

2. Cho một min-heap được biểu diễn dưới dạng array như sau: 1, 4, 7, 8, 10, 12, 13 Đâu là một biểu diễn ĐÚNG cho trạng thái của minheap sau khi xóa một phần tử ra khỏi min-heap

   a) 4, 7, 8, 10, 13, 12
   b) 4, 7, 8, 13, 10, 12
   c) 4, 8, 13, 7, 10, 12
   d) 4, 8, 7, 13, 10, 12

3. Đâu là độ phức tạp của thao tác ReHeapUp?

   a) O(logn) b) O(n) c) O(nlogn) d) O(1)

4. Đâu là độ phức tạp của thao tác build heap?

   a) O(logn) b) O(n) c) O(nlogn) d) O(1)

5. Đâu là độ phức tạp của heap sort?

   a) O(logn) b) O(n) c) O(nlogn) d) O(1)

6. Đâu là độ phức tạp của thao tác xóa 1 phần tử?

   a) O(logn) b) O(n) c) O(nlogn) d) O(1)

7. Đâu là độ phức tạp của giải thuật Heapsort ở trường hợp tốt nhất (best-case)

   a) O(logn) b) O(n) c) O(nlogn) d) $O(n^2)$

8. Cho max-heap mang các giá trị số nguyên như sau [29, 20, 10, 15, 18, 9, 5, 13, 2, 4, 15]. Lần lượt lấy m phần tử từ đỉnh của max-heap nói trên để tổng của m phần tử này không vượt quá 70. Hãy cho biết giá trị của m.

   a) 2 b) 3 c) 4 d) 5

9. Với hai max heap có kích thước n, độ phức tạp thời gian tối thiểu có thể có để tạo một maxheap từ các phần tử của hai maxheap là bao nhiêu?

   a) O(logn) b) O(n) c) O(nlogn) d) $O(n^2)$

10. Trong một maxheap nhị phân chứa n phần tử, phần từ nhỏ nhất có thể được tìm thấy trong thời gian

    a) O(logn) b) O(n) c) O(nlogn) d) $O(n^2)$

11. Một min-heap nhị phân hoàn chỉnh được tạo bằng cách bao gồm mỗi số nguyên trong [1, 1023] đúng một lần. Độ sâu của một nút trong heap là độ dài của đường dẫn từ gốc của heap đến nút đó. Do đó, gốc ở độ sâu 0. Độ sâu tối đa mà số nguyên 9 có thể xuất hiện là

    a) 6 b) 7 c) 8 d) 9

12. Chèn thêm 7, 2, 10 và 4 vào 3-ary max heap 9, 5, 6, 8, 3, 1. Max heap sẽ trở thành

    a) 10, 8, 6, 9, 7, 2, 3, 4, 1, 5
    b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
    c) 10, 7, 9, 8, 3, 1, 5, 2, 6, 4
    d) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3

13. Cho một mảng gồm 7 biến $[x_0, \ldots, x_6]$. Giả sử sau quá trình chuyển mảng trên thành min-heap ta được mảng: $[x_6, x_4, x_2, x_3, x_1, x_5, x_0]$. Mảng nào sau đây có thể là kết quả sắp xếp tăng dần của mảng trên?

    a) $[x_6, x_1, x_2, x_3, x_4, x_5, x_0]$
    b) $[x_6, x_2, x_5, x_0, x_3, x_4, x_1]$
    c) $[x_6, x_4, x_3, x_2, x_5, x_0, x_1]$
    d) $[x_6, x_4, x_1, x_3, x_5, x_2, x_0]$

14. Cho max-heap: $90, 73, 41, 25, 36, 17, 1, 2, 3, 19, 26, 7$. Trạng thái heap sau khi xoá 36 khỏi heap:

a) $90, 73, 41, 25, 26, 17, 1, 2, 3, 7, 19$
b) $90, 73, 41, 25, 19, 17, 1, 2, 3, 7, 26$
c) $90, 73, 41, 25, 26, 17, 1, 2, 3, 19, 7$
d) $90, 73, 41, 25, 19, 17, 1, 2, 3, 26, 7$
e) Không thể xoá node trung gian trong heap

15. Cho một mảng: $1, 3, 5, 4, 6, 11, 10, 8, 7, 13, 15$. Đâu là mảng đầu ra đại diện cho kết quả sau hai bước heapify mảng trên thành max heap với giải thuật $O(N)$?

a) $[15, 13, 11, 8, 6, 5, 10, 4, 7, 3, 1]$
b) $[1, 15, 11, 8, 13, 5, 10, 4, 7, 3, 6]$
c) $[1, 3, 5, 8, 15, 11, 10, 4, 7, 13, 6]$
d) Các đáp án khác đều sai

16. Mảng nào sau đây biểu diễn max heap?

a) $[17, 15, 13, 9, 6, 10, 5, 4, 8, 3, 1]$
b) $[17, 15, 13, 9, 6, 5, 10, 4, 8, 3, 1]$
c) $[17, 15, 13, 9, 6, 5, 4, 10, 3, 8, 1]$
d) $[17, 13, 15, 6, 9, 4, 10, 5, 3, 1, 8]$

17. Cho một mảng: $[10, 7, 11, 5, 4, 13, 1, 2]$. Đâu là mảng đầu ra đại diện cho min heap được tạo ra bằng cách thêm vào từng phần tử của mảng trên?

a) $[1, 2, 4, 5, 7, 13, 11, 10]$
b) $[1, 2, 5, 4, 7, 13, 11, 10]$
c) $[1, 2, 5, 4, 7, 11, 13, 10]$
d) $[1, 2, 4, 5, 7, 10, 11, 13]$

18. Cho min-heap: $[1, 2, 6, 4, 7, 13, 11, 10]$. Đâu là min-heap đầu ra nếu ta thêm vào mảng trên phần tử có giá trị 3?

a) $[1, 2, 3, 4, 7, 13, 11, 10, 6]$
b) $[1, 2, 6, 3, 7, 13, 11, 10, 4]$
c) $[1, 2, 6, 4, 7, 13, 11, 10, 3]$
d) $[1, 2, 6, 4, 3, 13, 11, 10, 7]$

19. Phát biểu nào sau đây là đúng với một heap có độ cao h:

a) Phần tử có key lớn nhất sẽ là root của heap
b) Các phần tử leaf của heap chỉ nằm ở độ cao h
c) Heap là một cây nhị phân hoàn chỉnh (complete binary tree)
d) Các phát biểu khác đều đúng

20. Độ phức tạp của các giải thuật xây dựng heap như heapify và bottom-up-build tương ứng là:

a) $O(\log_2 n)$ và $O(n)$
b) $O(n)$ và $O(\log_2 n)$
c) $O(n)$ và $O(n)$
d) $O(\log_2 n)$ và $O(\log_2 n)$
e) Tất cả các câu trên đều sai

21. Với chức năng trả về vị trí của đỉnh có trọng số nhỏ nhất và sau đó loại bỏ đỉnh này khỏi danh sách ở mỗi lần lặp, nên áp dụng cấu trúc dữ liệu nào cho hàm minDistance thì thích hợp nhất để giảm độ phức tạp thời gian thực thi?

a) Đống (Heap)
b) Chồng (Stack)
c) Danh sách liên kết đơn (Singly Linked List)
d) AVL

22. Cho một 3-ary max heap: $27, 21, 15, 18, 19, 11, 9, 10, 12, 14, 8$. Trạng thái heap sau khi xoá phần tử 19 là:

a) $27, 21, 15, 18, 14, 11, 9, 10, 12, 8$
b) $27, 21, 15, 18, 12, 11, 9, 10, 14, 8$
c) $27, 21, 15, 18, 8, 11, 9, 10, 14, 12$
d) Không thể xoá phần tử trung gian trong heap

23. Cho một 3-ary max heap ban đầu: $30, 20, 15, 18, 19, 11, 10$. Trạng thái heap sau khi chèn thêm các phần tử 25 và 5 là:

a) $30, 25, 15, 20, 19, 11, 10, 18, 5$
b) $30, 25, 15, 18, 19, 11, 10, 20, 5$
c) $30, 20, 25, 18, 19, 11, 10, 15, 5$
d) $30, 20, 15, 18, 19, 11, 10, 25, 5$

Dữ liệu sau cho 2 câu tiếp theo:

```
class Heap {
    int *harr; // pointer to array of elements in heap
    int heap_size; // Current number of elements.
public:
    bool isMinHeap() {
        // This function will return true if given level order
        ↪ traversal is Min Heap
        for (int i = (heap_size / 2 - 1); i >= 0; i--) {
            if (/*Code1*/)
                return false;
            if (/*Code2*/) {
                if (harr[i] > harr[2 * i + 2])
                    return false;
            }
        }
        return true;
    }
};
```

24. Hãy điền vào `Code1`:

a) harr[i] < harr[2 * i + 1]
b) harr[i] > harr[2 * i + 2]
c) harr[i] < harr[2 * i + 2]
d) harr[i] > harr[2 * i + 1]

25. Hãy điền vào `Code2`:

a) 2*i + 1 < heap_size
b) 2*i + 2 < heap_size
c) 2*i + 2 <= heap_size
d) 2*i + 1 == heap_size

Dữ liệu sau cho 2 câu tiếp theo:

```
class Heap {
    int *harr; // pointer to array of elements in heap
    int heap_size; // Current number of elements.
public:
    void buildHeap() {
        // This function will build a heap from the array
        for (int i = (heap_size / 2 - 1); i >= 0; i--) {
            heapify(i);
        }
    }

    void heapify(int i) {
        int left = 2 * i + 1;
```

```
        int right = 2 * i + 2;
        int smallest = i;

        if (left < heap_size && /*Code1*/)
            smallest = left;
        if (right < heap_size && harr[right] < harr[smallest])
            smallest = right;
        if (smallest != i) {
            std::swap(harr[i], harr[smallest]);
            /*Code2*/
        }
    }
};
```

26. Hãy điền vào `Code1`:

a) harrharr[left] < harr[smallest]
b) harr[left] < harr[smallest + 1]
c) harr[left] > harr[smallest]
d) harr[left] > harr[smallest + 1]

27. Hãy điền vào `Code2`:

a) heapify(left);
b) heapify(right);
c) heapify(smallest);
d) heapify(smallest - 1);

Dữ liệu sau cho 2 câu tiếp theo:

```
class Heap {
    int *harr; // pointer to array of elements in heap
    int heap_size; // Current number of elements.
public:
    void buildHeap() {
        // This function will build a heap from the array using
        ↪ reheap up
        for (int i = 1; i < heap_size; i++) {
            reheapUp(i);
        }
    }

    void reheapUp(int i) {
        int parent = /*Code2*/;

        for (; i > 0 && /*Code1*/; i = parent) {
            std::swap(harr[i], harr[parent]);
            parent = (i - 1) / 2;
        }
    }
};
```

28. Hãy điền vào `Code1`:

a) harr[parent] > harr[i + 1]
b) harr[parent] > harr[i - 1]
c) harr[parent] < harr[i - 1]
d) harr[parent] < harr[i + 1]

29. Hãy điền vào Code2:

a) (i - 1) / 2
b) (i + 1) / 2
c) i / 2
d) i - 1

Dữ liệu sau cho 2 câu tiếp theo:

```
class Heap {
    int *harr; // pointer to array of elements in heap
    int heap_size; // Current number of elements.

public:
    // Constructor to initialize heap
    Heap(int arr[], int n) {
        // TODO
    }

    void heapSort() {
        // Build a max heap
        for (int i = /*Code1*/; i >= 0; i--) {
            heapify(i);
        }

        // Extract elements from heap one by one
        for (int i = heap_size - 1; i > 0; i--) {
            std::swap(harr[0], harr[i]);
            heap_size--;
             /*Code2*/
        }
    }

    // To maintain max-heap property
    void heapify(int i) {
         // TODO
    }
};
```

30. Hãy điền vào Code1:

a) heap_size / 2 - 1
b) heap_size + 1
c) heap_size / 2 - 2
d) heap_size * 2

31. Hãy điền vào Code2:

a) heapify(0);
b) heapify(1);
c) heapify(i);
d) heapify(i-1);