2 4 6 7 9 10 🡸🡺 2 3 4 6 7 8 9 🡪 2 2 3 4 4 6 6 7 7 8 9 9 10

Merge Sort.

Chia một mảng lớn ra thành ra 2 mảng nhỏ sort 2 mảng nhỏ đó

Sau đó ta merge 2 mảng nhỏ đó thành một mảng lớn 🡪 mảng lớn đã được sort

Chia đôi mảng nửa bên trái thành 2 mảng, tương tự như vây ta chia đôi mảng bên phải

Vd: 2 4 5 1 3 5 7 4 2 4 5 6 7 9

Bước chia đôi mảng

2 4 5 1 3 5 7 🡨🡪 4 2 4 5 6 7 9

Lại chia đôi mảng con thành 2 mảng nhỏ nữa

2 4 5 1 <-> 3 5 7 🡨 🡪 4 2 4 5 <->6 7 9

2 4 - 5 1 <-> 3 5 - 7 🡨 🡪 4 2 - 4 5 <->6 7 -9

Chia đã xong, merge:

2 4 - 1 5 <-> 3 5 - 7 🡨 🡪 2 4 - 4 5 <->6 7 -9

Merge:

2 4 - 1 5 <-> 3 5 - 7 🡨 🡪 2 4 - 4 5 <->6 7 -9

1 2 4 5 <->3 5 7 🡨🡪2 4 4 5<-> 6 7 9

Merge:

1 2 3 4 5 5 7 🡨🡪 2 4 4 5 6 7 9

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

44 32 68 43 65 24 50 57 24 55 92 81 98 97 31 52

1 2 4 5 <->3 5 7

b = 0 e =7, m =4

c[7], i =b=0,j =4 ;i<m;j<e

c[0] =1, i =1 ,k=1

i=1<4; j=4<7

c[1] =2 i =2, k=2

i=2<4;j =4<7 v

c[2] =3 ;j =5 ; k =3

i=2<4;j=5<7 v

a[2]<a[5]

1 2 4 5 <->3 5 7

c[3] =4, i = 3;k=4

i=3<4;j=5<7 v

a[3]<a[5]

c[4] =5; j=6;k = 5

i = 3<4 j=6<7 v

a[3]<a[6] (5<7)v

c[5] = 5; i =4 j=6; k=6

i=4<4 j =6<7 sai vòng while sẽ dừng

i=4==m

Bắt đầu thực hiện vòng for

J=6<7 v

c[6] =7

k=7;j=7

j=7<7 sai🡪 vòng for sẽ dừng

52 79 20 33 55 20 22 25

Sort(0,8)

0>=8 sai

8<=0+2 sai

M =(8+0)/2 = 4

Sort(a,0,4) (1)

Sort(a,4,8) (2)

Sort(1)

0>=4 sai

4<=2+0 sai

m(1) = 2

sort(a,0,2)(1.1)

sort(a,2,4)(1.2)

sort(a,0,2)

0>=2 sai

2<=2+0 true

52>79 sai ko lam gì sort(1.1)

Sort(1.2) ko lam gi de nguyen

52 79 20 33

20 33 52 79 (1)

Sort(2)

B =4 e=8

M =6

Sort(a,4,6)(2.1) 🡪 20 55

Sort(a,6,8) (2.2) 🡪22 25

Sort (2)

20 22 25 55

20 33 52 79 ⬄ 20 22 25 55

**BTVN: Viết ra giấy merge sort 8 số ngẫu nhiên.**

Heap sort

Cây heap

Cây heap: nó là một cây nhị phân cân bằng và gần đầu đủ có thêm điều kiện \*

Cây nhị phân cân bằng: là cây nhị phân mà độ cao của con bên trái và độ cao của con bên phải của một node bất kỳ không lệch quá 1 đơn vị

Thế nào là cây nhị phân cân bằng gần đủ: nó sẽ lấp đầy từ trên xuống dưới và từ trái qua phải

Nếu chiều cao của cây nhị phân là n

Mà số node là 2n-1 người ta gọi là cây cân bằng tuyệt đối (perfect balance tree)

Cây heap: nó là một cây nhị phân cân bằng và gần đầu đủ có thêm điều kiện với node bất kỳ giá trị luôn lớn hơn giá trị 2 con của nó (max heap) nếu node bất kỳ có giá trị nhỏ hơn giá trị 2 con của nó (min heap).

Với cây heap ko quan tâm đến so sánh giữa giá trị con trái và con phải, mà người ta chỉ quan tâm đến giá trị cha với các giá trị con của nó mà thôi

Đối với cây BSTree: left<parent<right

Đối với cây Head: parent >=left, && parent>= right

Left 🡪 parent 🡪 right (parent nằm giữa 2 con )(BSTree)

Parent (left, right): min heap,

(left, right) parent: max heap

|  |  |
| --- | --- |
| Đối với cây BSTree: left<parent<right | Đối với cây Head: parent >=left, && parent>= right |
| Left 🡪 parent 🡪 right (parent nằm giữa 2 con )(BSTree) | Parent <=(left, right): min heap,  (left, right) <=parent: max heap |
| BSTree các giá trị là key (duy nhất không lặp lại) | Heap: các giá trị có thể duplicate |
| Chỉ cần là cây nhị phân | Cây nhị phân cân bằng và gần đủ |
| Sinh ra để tìm kiếm | Dùng để sort, dùng queue ưu tiên (dùng computer science) |

Cây heap có thể diễn tả bởi mảng

Giả sử một cây heap đã được xây dựng thì khi chúng ta duyệt (breadth first) cây heap nó sẽ tạo ra một dãy số giống nhưng một mảng

Với một node có chỉ số là i:

Cha của node i: f =(i-1)/2

Con bên trái của node i: CL =2\*i+1

Con bên phải của node i: CR = 2\*i+2

Biểu diễn mảng thành cây heap:

A,b,c,d,e,f,g,h,i,j

A

B C

D E F G

H I J ….

Build cây heap

52 79 20 33 55 20 22 25

Có 2 loại xây dựng cây heap từ mảng:

Loại 1: top down (đi từ trên xuống dưới)

Loại 2: từ dưới lên:

Nguyên tắc của loại 1:

Đưa phần tử đầu tiên vào mảng

52

Sau cứ mỗi phần tử ta đưa vào và kiểm tra xem nó có nhỏ hơn cha của nó ko?

Nếu nhỏ hơn hoặc bằng thì ta thôi, còn nếu nó lớn hơn cha ta đổi chỗ phần tử đó cho cha của nó, cứ là như thế cho đến khi ko phải đổi chỗ nưa thì thôi.

52 79 🡪 79 52

Đừa 20🡪 79 52 20

52 79 20 33 55 20 22 25

Đưa 33 (i=3) 79 52 20 33

Đưa 55 (i=4) 79 55 20 33 52 (i=1)🡪 so 55 với 79

Đưa 20 vào (i=5) 79 55 20 33 52 20

Đưa 22 vào (i=6) 79 55 22 33 52 20 20 (i=2)

Đưa 25 (i=7) 79 55 22 33 52 20 20 25

79

55 22

33 52 20 20

25

Loại 1: cách 2 đơn giản dùng draw:

52 79 20 33 55 20 22 25

79

55 22

33 52 20 20

25

Loại 2: bottom 🡪 up

Nguyên tắc: Xuất phát từ thằng cuối cùng sau đó xét (nửa cuối của mảng) lên đến gốc xem nó có thỏa mãn điều kiện con<=cha, nếu ko thỏa mản thì ta đổi cho con cho cha

52 79 20 33 55 20 22 25

52 79 20 33 55 20 22 25

52 79 20 33 55 20 22 25 🡪 79 52 20 33 55 20 22 25

79 52 20 33 55 20 22 25 🡪 79 52 22 33 55 20 20 25

79 52 22 33 55 20 20 25

79 52 22 33 55 20 20 25

79 52 22 33 55 20 20 25

79 52 22 33 55 20 20 25 🡪79 55 22 33 52 20 20 25

79 55 22 33 52 20 20 25

79

55 22

33 52 20 20

25

Enqueue 🡪 Khi Enqueue vẫn phải đảm bảo đó là cây heap

Enqueue cây 100

79

55 22

33 52 20 20

25 100

Sau đó reheap lại cây ứng với giá trị vừa đưa vào

100

79 22

55 52 20 20

25 33

Enqueue (85)

100

79 22

55 52 20 20

25 33 85

Reheap

100

85 22

55 79 20 20

25 33 52

Dequeue:

Khác với queue thông thường (FIFO) các phần tử mà có trọng số lớn sẽ được ưu tiên dequeue trước

Dequeue root

52

85 22

55 79 20 20

25 33

Reheap:

85

52 22

55 79 20 20

25 33

85

79 22

55 52 20 20

25 33

Dequeue 🡪 85

33

79 22

55 52 20 20

25

Reheap:

79

55 22

33 52 20 20

25

Dequeue 🡪 79

25

55 22

33 52 20 20

Reheap:

55

25 22

33 52 20 20

55

52 22

33 25 20 20

Head Sort:

100

85 22

55 79 20 20

25 33 52

Đổi phần tử root về vị trị cuối cùng của các phần tử cần xét

52

85 22

55 79 20 20

25 33 100

Reheap nhưng lưu ý ko được reheap phần tử cuối cùng:

52

85 22

55 79 20 20

25 33 100

85

52 22

55 79 20 20

25 33 100

85

79 22

55 52 20 20

25 33 100

Đổi root phần tử cuối cùng

33

79 22

55 52 20 20

25 85 100

Reheap:

79

55 22

33 52 20 20

25 85 100

Đưa 79 về cuối cùng

25

55 22

33 52 20 20

79 85 100

Reheap:

55

52 22

33 25 20 20

79 85 100

Đưa 55 về cuối:

52

33 22

20 25 20 55

79 85 100

Đưa 52 về cuối

33

25 22

20 20 52 55

79 85 100

Đưa 33 về cuối

25

20 22

20 33 52 55

79 85 100

Đưa 25 về cuối

22

20 20

25 33 52 55

79 85 100

Đưa 22 về cuối

20

20 22

25 33 52 55

79 85 100

Đưa 20 về cuối

20

20 22

25 33 52 55

79 85 100

20 20 22 25 33 52 55 79 85 100

BTVN:

Sinh ra cho thầy 10 số bất kỳ: build thành cây heap theo 2 dạng Top-down và bottom – up

Cũng 10 số đó viết các bước để cần dùng cho heap sort