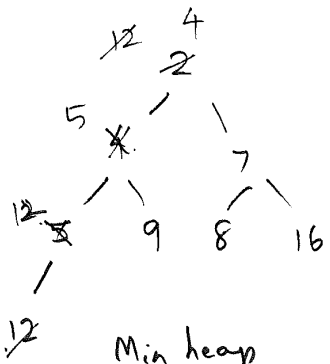
堆排序就是把堆顶的最大数取出，

将剩余的堆继续调整为最大堆，具体过程在第二块有介绍，以递归实现

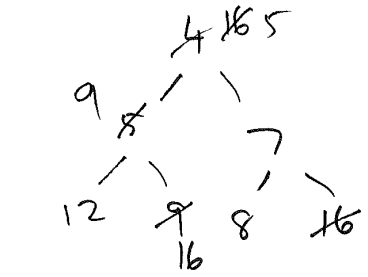
剩余部分调整为最大堆后，再次将堆顶的最大数取出，再将剩余部分调整为最大堆，这个过程持续到剩余数只有一个时结束

Heap sort



Min heap

remove( ) 2



remove( ) 4

heapsort

buildHeap(A) n ← A.length

for i ← n-1 downto 0 do

A[i] ← remove( )

2 4 7 5 9 8 16 12

Array rep

4 5 7 12 9 8 16 2

5 9 7 12 16 8 4 2

:

heap is empty

A is in descending order

Min heap → descending order

Max heap → ascending order

buildHeap(A): // Build a heap from an array (in place)

// Bottom-up heap building: 2

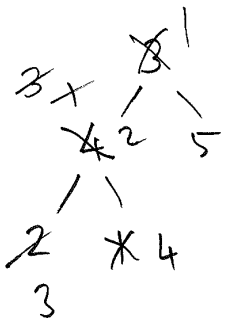
for i ← ⌊(A.length − 2)/2⌋ // ← last node with a child

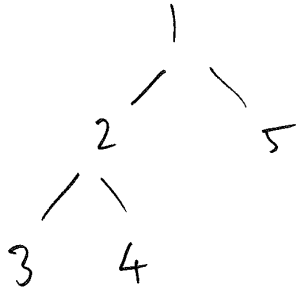
downto 0 do

percolateDown(i)

RT: O(n)

Example: A = 3 4 5 2 1





buildHeap:

for i ← parent(size − 1) downto 0 do

percolateDown(i)

Minimize error in computing floating point sum of $\sum_{i=1}^{n} a_i$

- Create a PQ with $a_1 \ldots a_n$
- while pq has more than one element do

```
x ← pq.remove()
y ← pq.remove()
pq.add(x + y)
```

- PQ has one element = sum.

Generating perfect powers:

pq of triplets: $\langle a, b, a^b \rangle$ priority $= a^b$

Initially add: $\langle 2, 2, 4 \rangle$.

Repeat:

remove $\langle a, b, a^b \rangle$ → output a^b (if not output already)

if $a = 2$: add $\langle 2, b+1, 2^{b+1} \rangle$ and $\langle 3, b, 3^b \rangle$ into pq.

else add $\langle a+1, b, (a+1)^b \rangle$ into pq.

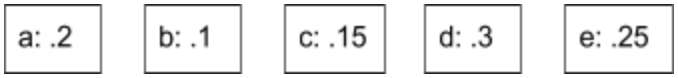
Applications of priority queues:
Huffman coding
Prim's algorithm for MST
Dijkstra's algorithm for shortest paths
Select algorithm (k largest elements of a large array or stream)
Process scheduling, interrupt handling in operating systems
Heapsort
Heuristics for memory management, bin packing
Discrete event simulations, computer games: (randomly generated) events are processed in temporal order
A* search in AI
Reduce round-off errors in floating point computations: best way to find sum of A[i] , i = 1..n? Is $1 + \varepsilon = 1$?
Merge sort using k-way merge
Find perfect powers (numbers of the form a^b) in increasing order, up to some n (say, 10^{18}).
Enumerate numbers whose prime factors are only from a given set (e.g., {3,5,7}), in sorted order.
Pairing buy/sell orders in the stock market by market makers

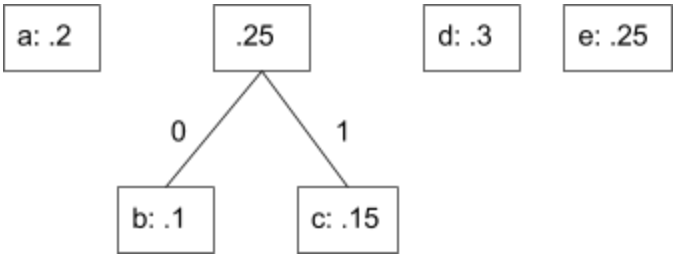
Huffman coding (application of priority queues):
Input: Alphabet Σ, and a frequency function $f : \Sigma \rightarrow \Re^+$. Example: Σ = {A, C, G, T}, f = {.5, .25, .1, .15}. In the given input file with n characters, $c \in \Sigma$ occurs $f(c) \cdot n$ times.
Output: A binary code for Σ that minimizes the total length of the file. It is required that no character's code is a proper prefix of another character's code. Such codes are called *prefix* codes, and no boundary markers are needed when a file is encoded with prefix codes.

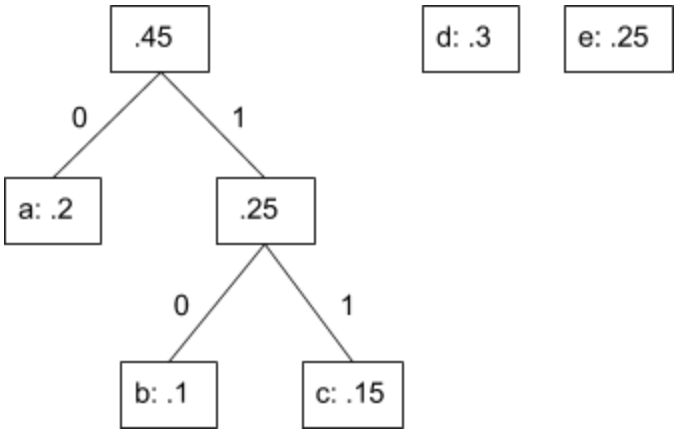
	A (0.5)	C (0.25)	G (0.1)	T (0.15)	Weighted average of bits per character
Fixed length code	00	01	10	11	2*.5+2*.25+2*.1+2*.15 = 2
Variable length code	0	10	110	111	1*.5+2*.25+3*.1+3*.15 = 1.75

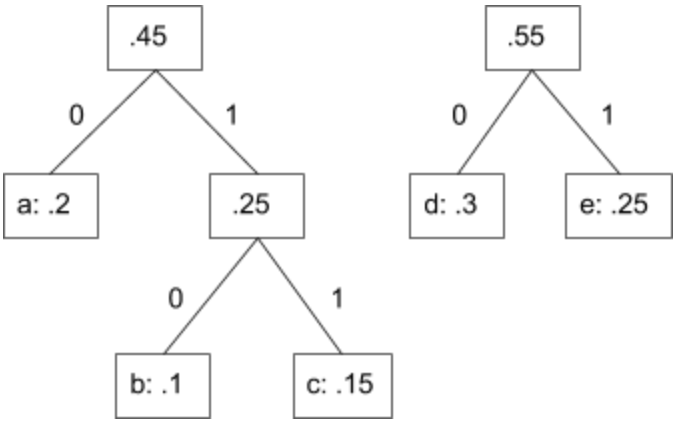
Huffman's algorithm to compute an optimal prefix code:
Create a priority queue q with the characters of Σ, where $c \in \Sigma$ has priority $f(c)$.
while q has more than one node do
    x ← q.remove( )
    y ← q.remove( )
    Create a new node z with frequency f(z) = f(x) + f(y).
    Attach nodes x and y as the 2 children of z, along edges labeled 0 and 1.
    q.add(z)
The single node in q contains a tree that is an optimal coding tree for the given problem.
Code for $c \in \Sigma$ is obtained by concatenating the labels along the edges from the root of the tree to the leaf node corresponding to c.

Huffman coding example: Σ = {a, b, c, d, e}, f = {.2, .1, .15, .3, .25}

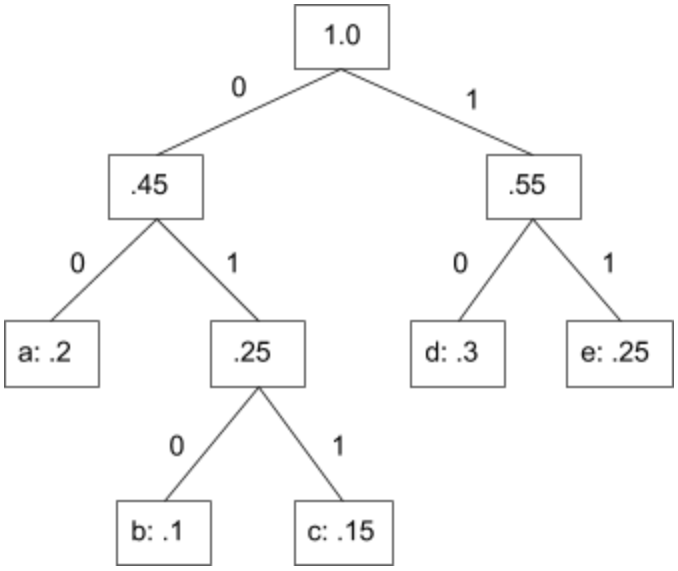








Final encoding tree:



Codes:

a	00
b	010
c	011
d	10
e	11

Weighted average number of bits per character:
2*.2 + 3*.1 + 3*.15 + 2*.3 + 2*.25 = 2.25.