ESERCIZIO 2

$P(a) = 0,5 = \frac{1}{2}$

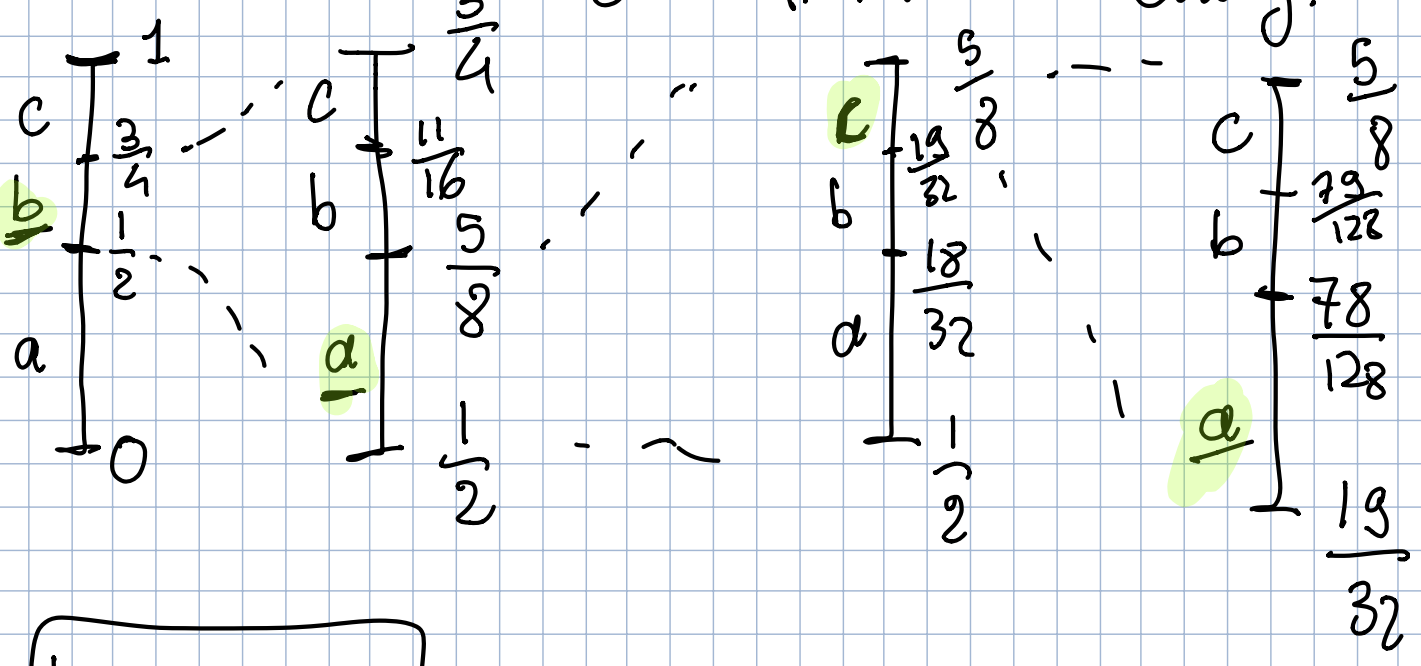
$P(b) = P(c) = 0,25 = \frac{1}{4}$

10011

10011 deve essere trasformata in una frazione diadica: $\frac{V}{2^k}$

$k = 5 \qquad V = 1 + 2 + 16 = 19$

$< \frac{19}{32} , 4 >$ Decodificare questo che è stato codificato con Arithmetic coding.



baca

ESERCIZIO 1

Lo Snow Plow è una tecnica inventata da Knuth in cui si cerca virtualmente di aumentare la dimensione della memoria M in modo da produrre dei run ordinati per il Merge Sort di dimensione in media $2M$ e non M.
Quindi questo permette di avere come complessità un numero di I/O:

$$O\left(\frac{n}{B}\log_2 \frac{n}{2M}\right) \text{ invece di } O\left(\frac{n}{B}\log_2 \frac{n}{M}\right)$$

In M creiamo un Min Heap.
L'algoritmo funziona così:
In memoria abbiamo M elementi.
→ Estraiamo il minimo dello heap e lo mettiamo in output
→ Prendiamo un elemento da S se è < minimo lo mettiamo in un array V else in H
→ Terminiamo quando abbiamo V di dimensione M e da quello che

e in U ricreiamo lo heap.

Dato che prendiamo γ elementi da S e che H vanno in U: abbiamo che $\gamma - H$ vanno in H. In output ci vanno

$$\gamma - H + H = \gamma$$

La prob. di andare in H o in U è $\frac{1}{2}$

quindi abbiamo $H = \frac{1}{2}\gamma \rightarrow \gamma = 2H$

Quindi in output vanno in media $2H$ elementi.

Simulazione:

S = 3, 10, 1, 6, 2, 5, 3 $\quad$ H=2

H = 3, 10
U =

→ output = 3

H = 10
U = 1

→ output = 3, 10 $\quad$ 1° run ordinato

H = ∅

Rebuilt the heap

U = 1,6

H = 1,6
U = ϕ
→ output = 1

H = 2,6
U = ϕ
→ output = 1,2

H = 5,6
U = ϕ
→ output = 1, 2, 5

H = 6
U = 3
→ output = 1, 2, 5, 6

H = ~~3~~
U = ϕ
→ output = 3

Produco 3 run ordinati:

{3, 10}

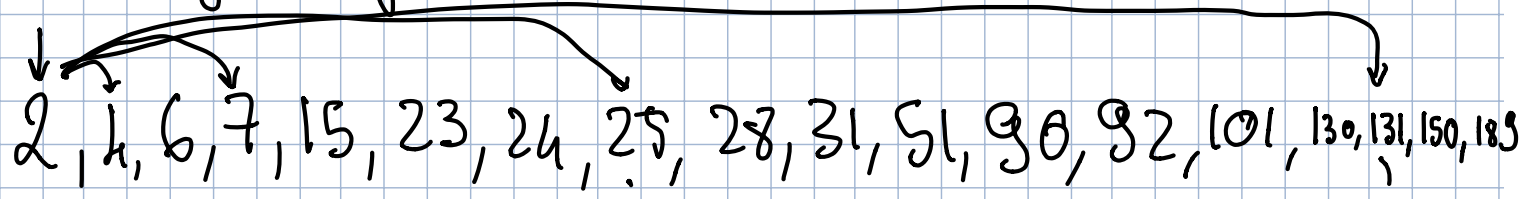
{1, 2, 5, 6}

{ 2

{3}

ESERCIZIO 6

Doubling algorithm: cerco 51:



$i = 0$ $k = 0$ → $2^k = 1$ $2 < 51$

$i = 0$ $k = 1$ → $2^k = 2$ $4 < 51$

$i = 0$ $k = 2$ → $2^k = 4$ $7 < 51$

$i = 0$ $k = 3$ → $2^3 = 8$ $25 < 51$

$i = 0$ $k = 4$ → $2^4 = 16$ $131 > 51$

Mi fermo e faccio la ricerca binaria tra $[8, 16]$ quindi tra 25 e 131 e trovo (51)

ESERCIZIO 3

Il quickSort non richiede spazio ausiliario come succede per il MergeSort, in realtà per le n chiamate ricorsive abbiamo una occupazione $O(n)$.

Vogliamo ridurre lo spazio necessario usando

une tecnica chiamata elimination of the tail recursion.

```
BandedQS (S, i, j)
  while (j - i > n0):
      r = pick a random pivot
      swap S[i] con S[r]
      p = partition (S, i, j)
      if (p < (j+i)/2)
          BandedQS (S, i, p-1)
          i = p+1
      else
          BandedQS (S, p+1, j)
          j = p-1
  InsertionSort (S)
```

l'idea è quella di ridurre il numero delle chiamate ricorsive svolgendo la chiamata a BandedQS sempre sulla parte più breve del sottoarray.

In questo modo riduciamo lo spazio da $O(n)$ a $O(\log n)$

$O(\lg n)$

ESERCIZIO 4

Dimostriamo che l'altezza della skip list è $O(\lg n)$ in media.
Abbiamo la lista con n elementi.
Per ognuno di questi abbiamo prob. $\frac{1}{2}$ di essere scelto e portato al livello superiore.
Se vogliamo considerare la probabilità che l'altezza L superi un valore l allora dobbiamo considerare che per l livelli almeno 1 nodo viene portato al livello superiore:

$$\sum_{i=1}^{l} \frac{1}{2^i} = \frac{1}{2^l}$$ dato che abbiamo n nodi allora diventa

$$\frac{n}{2^l} \quad \text{se } l = c \lg n \rightarrow \frac{n}{2^{c \lg n}}$$

$$\frac{n}{n^c} = \frac{1}{n^{c-1}} = O(\lg n)$$

ESERCIZIO 5