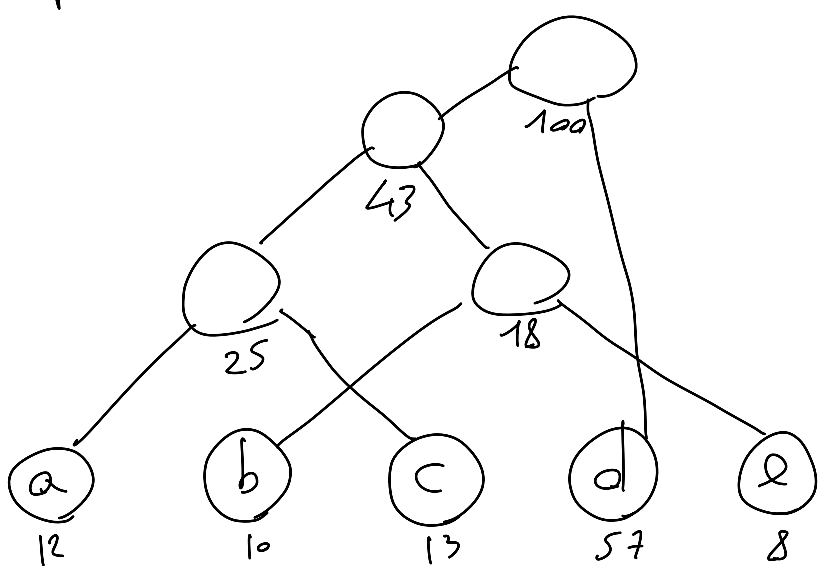
Domanda (5 punti) Indicare, in forma di albero binario, il codice prefisso ottenuto tramite l'alg. di Huffman per l'alfabeto $\{a, b, c, d, e\}$ con le seguenti frequenze

a	b	c	d	e
12	10	13	57	8

Spiegare il processo di costruzione del codice.



Esercizio (9 punti) Si consideri un file definito sull'alfabeto $\Sigma = \{a, b, c\}$ con frequenze $f(a), f(b), f(c)$. Per ognuna delle seguenti codifiche si determini, se esiste, un opportuno assegnamento di valori alle 3 frequenze $f(a), f(b), f(c)$ per cui l'algoritmo di Huffman restituisce tale codifica, oppure si argomenti che tale codifica non è mai ottenibile.

1) $e(a) = 0$ $e(b) = 10$ $e(c) = 11$

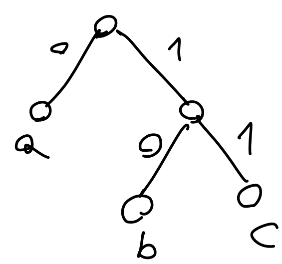
2) $e(a) = 1$ $e(b) = 0$ $e(c) = 11$

3) $e(a) = 10$ $e(b) = 01$ $e(c) = 00$

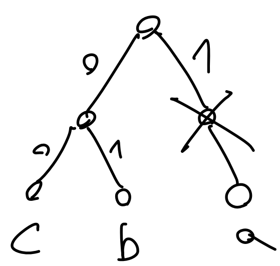
2) $e(a)$ è prefisso di $e(b)$, quindi questa codifica non è libera da prefissi, e quindi non è una codifica di Huffman

1) $f(a) > f(b), f(c)$

50 25 25

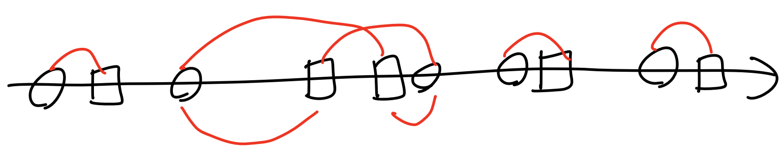


3)



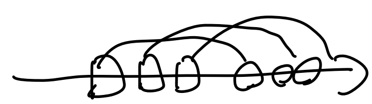
Esercizio: metric matching sulla linea

Sia $S = \{s_1, s_2, \dots, s_n\}$ un insieme di punti ordinati sulla retta reale, rappresentanti dei server. Sia $C = \{c_1, c_2, \dots, c_n\}$ un insieme di punti ordinati sulla retta reale, rappresentanti dei client. Il costo di assegnare un client c_i ad un server s_j è $|c_i - s_j|$. Si fornisca un algoritmo greedy che assegna ogni client ad un server distinto e che minimizzi il costo totale dell'assegnamento.

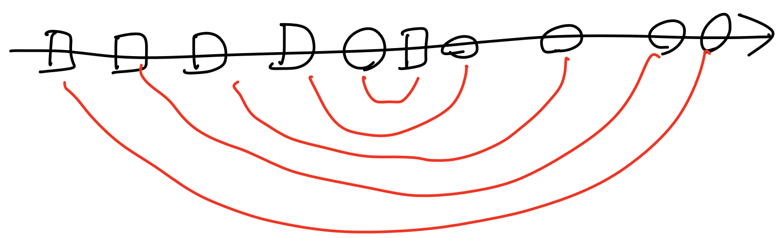


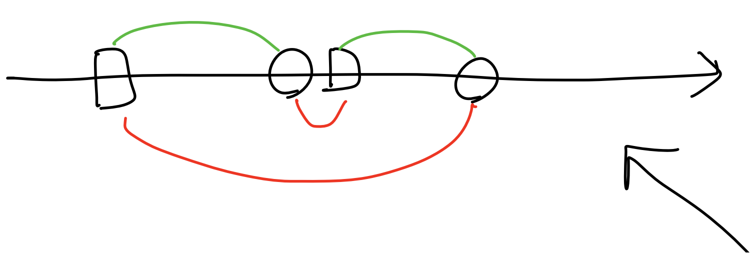
$n=5$

1) $c_1 - s_1$, $c_2 - s_2$, ...



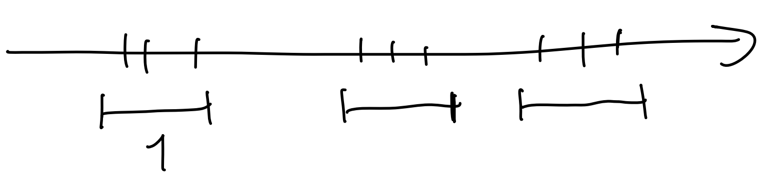
2)



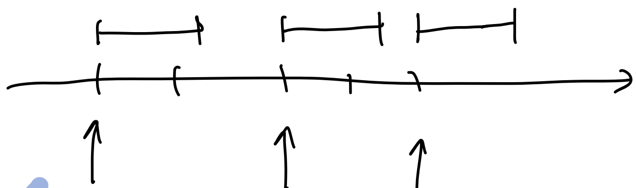


3) trova la coppia con distanza minore

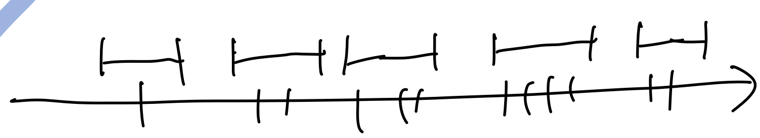
Esercizio: Sia $X = \{x_1, x_2, \dots, x_n\}$ un insieme di punti ordinati non decrescente sulla retta reale. Si fornisca un algoritmo greedy che determini un insieme I di cardinalità minima di intervalli chiusi di ampiezza unitaria ($[a,b] \in I \Rightarrow b-a=1$) tale che $\forall x_i \in X$ $\exists\, i \in I$ tale che $x_i \in i$.



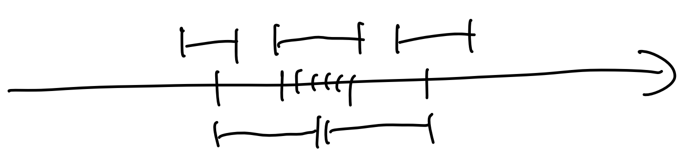
1)



2)



"gruppi più densi"



```
MIN_COVER(X)
  n ← length(X)
  C ← {[x_1, x_1+1]}
  last ← 1
  for i = 2 to n do
      if x_i > x_last + 1 then
         last ← i
         C ← {[x_i, x_i+1]} ∪ C
  return C
```