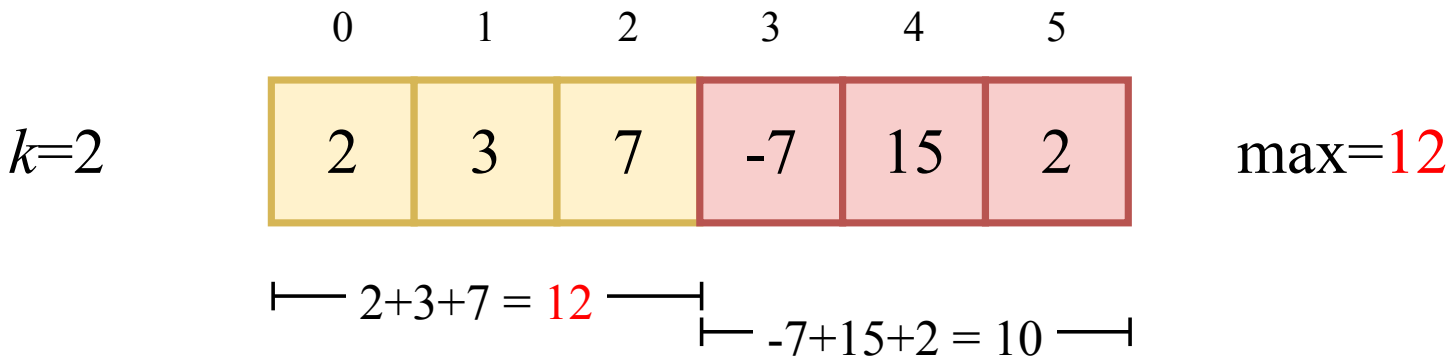
Supersequenza comune minimale

- Una stringa P è una supersequenza di una stringa T se T è una sottosequenza di P
- Scrivere un algoritmo che restituisca la lunghezza della supersequenza comune minimale di due stringhe P e T, cioè della più corta stringa che abbia sia P sia T come sottosequenze
- Discutere correttezza e complessità dell'algoritmo proposto
- Esempio 1: L'unica supersequenza comune minimale di `AB` e `BC` è `ABC`, quindi la sua lunghezza è 3
- Esempio 2: Le supersequenze comuni minimali di `DAB` e `DCB` sono `DACB` e `DCAB`, quindi la loro lunghezza è 4

OBIETTIVO: warm-up e riutilizzo di idee già viste in precedenza

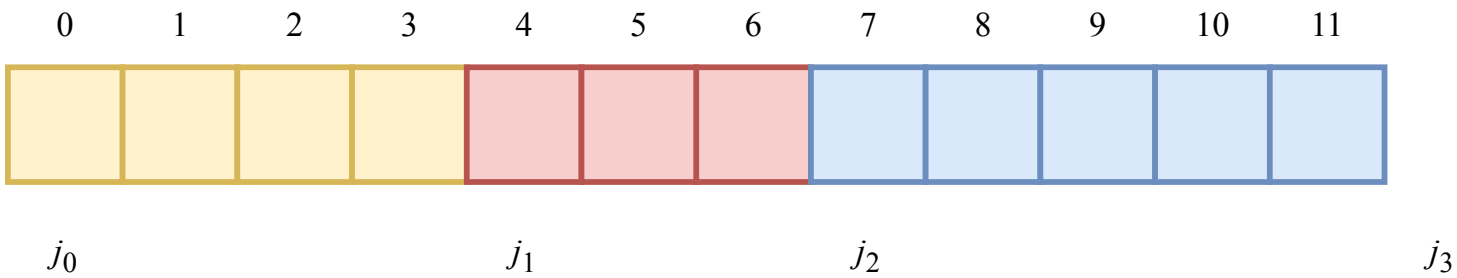
Costo partizione di un vettore

- Il costo $C(i, j)$ di un sottovettore $V[i \dots j]$ di V è pari alla somma dei suoi elementi
- Una k-partizione di V è una divisione di V in k sottovettori contigui e non vuoti che coprono totalmente il vettore e non si sovrappongono.
- Il costo della k-partizione è il costo massimo dei suoi sottovettori.



Costo partizione di un vettore (Versione formale)

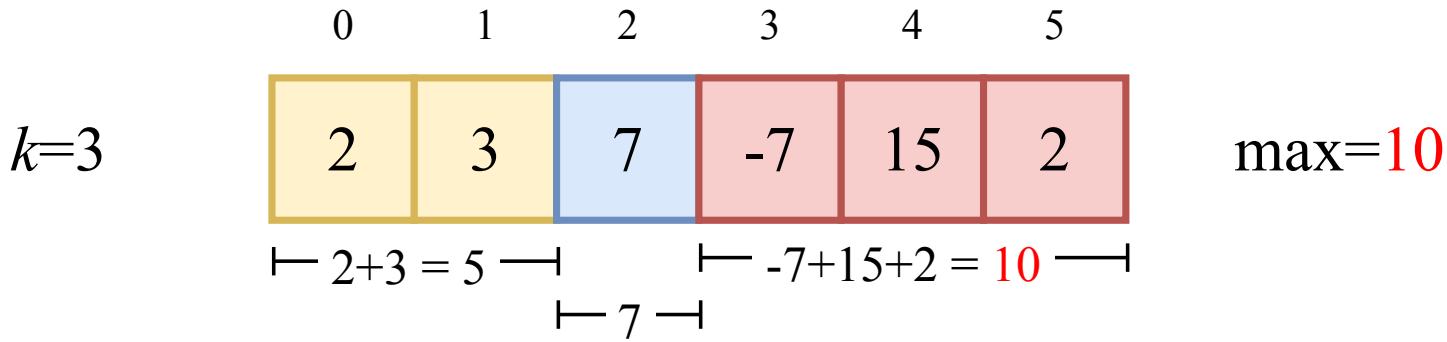
- Il costo di un sottovettore $V[i \dots j]$ è $C(i,j) = \sum_{t=i}^{j} V[t]$
- Una k-partizione di V è una divisione di V in k sottovettori contigui $V[j_0 \dots j_1 - 1], V[j_1 \dots j_2 - 1], \dots, V[j_{k-1} \dots j_k - 1]$ con $j_0 = 0, j_k = n$ e $j_t < j_{t+1}, \forall t, 0 \leq t < k$.
- Il costo della k-partizione è pari a:
$$\max\{C(j_t, j_{t+1} - 1) : \forall t, 0 \leq t \leq k - 1\}$$



SUGGERIMENTO: Verificate se la vostra comprensione del problema è in linea con la formulazione matematica dello stesso

Costo partizione di un vettore

Scrivere un algoritmo che prenda in input un vettore V contenente n interi e un intero k tale che $2 \leq k \leq n$ e restituisca il costo della k-partizione di V di costo minimo.



Strategia per approcciare il problema:

1. Soluzione naif per $k = 2$: $O(n^2)$; ottimizzata: $O(n)$
2. Soluzione naif per $k = 3$: $O(n^3)$; ottimizzata: $O(n^2)$
3. Soluzione naif per k: $O(n^k)$; ottimizzata $O(n^{k-1})$; tramite programmazione dinamica: $O(kn^2)$

Batterie (numero di fermate)

- Siete a bordo di un'auto elettrica su un'autostrada. Entrate in autostrada al km 0 con la batteria carica e dovete uscire al km L.
- Una batteria carica permette di percorrere R chilometri; prima che si esaurisca, bisogna fermarsi in un'area di servizio e sostituirla con una nuova batteria carica.
- Sia $dist$ un vettore strettamente crescente contenente n interi, dove $dist[i]$ è la distanza dell'area di servizio i-esima dal km 0.
- Scrivere un algoritmo che, dati $dist$, n, L e R, restituisca il numero minimo di fermate necessarie per arrivare al km L.
- Assumiamo che il viaggio sia possibile: la distanza fra 0 e $dist[0]$, ogni distanza fra due stazioni successive e la distanza fra $dist[n-1]$ e L sono minori o uguali a R; oppure, $L \leq R$.

Batterie (costo di sostituzione)

Consideriamo una variante del problema precedente, dove la sostituzione ha un costo. Per semplificare la formulazione, aggiungiamo una posizione fittizia finale che rappresenta l'uscita dall'autostrada.

- Sia *dist* un vettore strettamente crescente contenente $n+1$ interi positivi: per $0 \leq i < n$, $dist[i]$ è la distanza dell'area di servizio i-esima dal km 0, mentre $dist[n] = L$ è l'uscita dell'autostrada.
- Sia *cost* un vettore contenente $n+1$ interi non negativi: per $0 \leq i < n$, $cost[i]$ è il costo di una nuova batteria nell'area i-esima, mentre $cost[n] = 0$ indica che non è necessario pagare per arrivare a L.
- Il costo totale del viaggio è dato dalla somma dei costi delle batterie sostituite per arrivare al km L.
- Scrivere un algoritmo che, dati *dist*, *cost*, n e R, restituisca il costo totale minimo. Discutere correttezza e complessità.

Donald Trump

La Route 66 è una strada che collega Chicago a Los Angeles e lungo la quale si trovano n città. Nel suo tour elettorale Donald Trump (DJT) ha deciso di percorrerla in una sola direzione, senza mai tornare indietro sui propri passi. DJT non può tenere un comizio in ogni città, ma deve sceglierne un sottoinsieme.

Date due città i, j in cui terrà un comizio, esse devono trovarsi a distanza maggiore o uguale a D. La città i-esima si trova al miglio $miles[i]$; quindi la distanza fra i e j è pari a $|miles[j] - miles[i]|$.

Seguendo queste regole, The Donald vorrebbe parlare al maggior numero possibile di potenziali elettori. Si stima che al comizio nella città i-esima saranno presenti $people[i]$ persone.

Donald Trump

Sorprendentemente, The Donald non ha grandi conoscenze informatiche e ha chiesto a voi di risolvere il problema; in particolare, vorrebbe un algoritmo

int serveDJT(**int**[ ] *miles*, **int**[ ] *people*, **int** n, **int** D)

che prenda in input il vettore *miles* delle posizioni delle città (strettamente crescente), il vettore *people* del numero stimato di partecipanti, la dimensione n dei vettori e la distanza minima D tra due città in cui tenere un comizio, e restituisca il maggior numero complessivo di persone che possono essere presenti ai suoi comizi.

Thank you for your attention to this matter! DJT