Esercizio 1 Si consideri il seguente problema. Dato un vettore V di n interi positivi non superiori a b, trovare tutte le partizioni del vettore in sottovettori con la proprietà che la somma dei valori in ciascun sottovettore non supera il valore b. Ad esempio per $b = 5$ e $V = [2, 3, 3, 1]$ le partizioni possibili sono $\{[2], [3], [3], [1]\}$, $\{[2, 3], [3], [1]\}$, $\{[2, 3], [3, 1]\}$, $\{[2], [3], [3, 1]\}$.

1. *(max 10 punti)* Scrivere in pseudo–codice una procedura che risolve il problema. Una partizione è rappresentata dall'insieme degli indici degli ultimi elementi dei sottovettori che la compongono. Ad esempio per $b = 5$ e $V = [2, 3, 3, 1]$ la procedura deve stampare (non necessariamente in quest'ordine): $\{1, 2, 3, 4\}$, $\{2, 3, 4\}$, $\{2, 4\}$, $\{1, 2, 4\}$. La complessità della procedura **deve essere** $O(nD(n))$, dove $D(n)$ è il numero di partizioni da stampare.

2. *(max 5 punti)* Spiegare perché la procedura proposta ha la complessità richiesta.

Soluzione Esercizio 1

1. Un algoritmo che risolve il problema e che si basa sulla tecnica del backtracking è il seguente. L'algoritmo PARTIZIONE, essendo ricorsivo, prende in input gli interi n e b, il vettore V da partizionare, il vettore SOL in cui viene memorizzata la partizione da stampare, l'intero k rappresentante il numero di sottovettori nella partizione da stampare, il numero i di elementi inseriti nei sottovettori della partizione, il valore tot somma degli elementi di un eventuale sottovettore parziale presente nella partizione che si va costruendo. Quindi la prima chiamata sarà PARTIZIONE(n, b, V, SOL, 0, 0, 0).

```
PARTIZIONE:  INPUT gli interi n e b, i vettori V e SOL e gli interi k, i e tot
    IF i = n THEN stampa SOL[1], SOL[2], ..., SOL[k]
    ELSE
        SOL[k + 1] ← i + 1
        PARTIZIONE(n, b, V, SOL, k + 1, i + 1, 0)
        IF i + 1 < n AND tot + V[i + 1] + V[i + 2] ≤ b THEN
            PARTIZIONE(n, b, V, SOL, k, i + 1, tot + V[i + 1])
    ENDIF
```

2. Il contatore tot tiene traccia del valore dell'ultimo sottovettore parziale nella partizione che si va costruendo e viene aggiornato coerentemente alle scelte fatte: se si estende il sottovettore con l'elemento $V[i + 1]$ allora tot viene incrementato di $V[i + 1]$, altrimenti tot viene azzerato.
   Onde evitare la generazione di partizioni inutili, si procede come segue: il sottovettore viene esteso se e solo se $tot + V[i + 1] \leq b$ (in caso contrario si genererebbe un sottovettore la somma dei cui elementi eccede b) ed inoltre $i + 1 \neq n$ (in caso contrario il sottovettore comprendendo gia l'elemento n-mo del vettore non può essere esteso).

   Per quanto detto una chiamata ricorsiva sarà effettuata se e solo se la partizione parziale fino a quel momento costruita può estendersi ad una partizione da stampare. In questo modo la complessità risultante sarà $O(nD(n))$ dove $D(n)$ è il numero di partizioni da stampare.

Esercizio 2 Si consideri il seguente problema (variante del problema dell'esercizio 1). Dato un vettore V di n interi positivi non superiori a b, bisogna partizionarlo in sottovettori la somma dei cui valori non eccede il valore b, utilizzando il minor numero di sottovettori. Una partizione è rappresentata dall'insieme degli indici degli ultimi elementi dei sottovettori che la compongono. Ad esempio per $b = 5$ e $V = [2, 3, 3, 1]$ la partizione ottima è $\{2, 4\}$. Si propongono i seguenti algoritmi:

```
PART1:
INPUT gli interi b e n ed il vettore V
    tot ← 0
    SOL ← ∅
    i ← 1
    WHILE i < n DO
        IF tot + V[i] + V[i+1] > b THEN
            SOL ← SOL ∪ {i}
            tot ← 0
        ELSE tot ← tot + V[i]
        i ← i + 1
    ENDWHILE
    SOL ← SOL ∪ {n}
OUTPUT SOL.
```

```
PART2:
INPUT gli interi b, i, j e il vettore V
    IF i > j THEN SOL ← ∅
    ELSE
        Trova un sottovettore di V[i, j]
        con somma non eccedente b di
        massima lunghezza, e siano s e t gli
        estremi di tale sottovettore.
        SOL ← PART2(b, i, s − 1, V) ∪ {t} ∪
                ∪ PART2(b, t + 1, j, V)
    ENDIF
OUTPUT SOL.

La chiamata iniziale è PART2(b, 1, n, V).
```

1. *(max 10 punti)* Dire se l'algoritmo PART1 risolve il problema e in caso affermativo dimostrare la correttezza dell'algoritmo mentre in caso negativo fornire un controesempio.

2. *(max 10 punti)* Dire se l'algoritmo PART2 risolve il problema e in caso affermativo dimostrare la correttezza dell'algoritmo mentre in caso negativo fornire un controesempio.

Soluzione Esercizio 2

1. L'algoritmo PART1 risolve il problema.
   Per provare che l'algoritmo è corretto è sufficiente dimostrare che ogni nuova estensione della soluzione parziale mantiene la proprietà di essere contenuta in una soluzione ottima. Sia SOL_i l'insieme dei primi i indici inseriti nella soluzione dall'algoritmo. .

   $$\textit{Per ogni } i \geq 0 \textit{ esiste una soluzione ottima } SOL^* \textit{ tale che } SOL_i \subseteq SOL^*.$$

   Dimostrazione: per induzione su i.
   Poichè $SOL_0 = \emptyset$, per $i = 0$ è banalmente vero. Supponiamo ora che sia vero per i e dimostriamolo per $i + 1$. Per ipotesi induttiva esiste una soluzione ottima SOL^* tale che $SOL_i \subseteq SOL^*$. Se vale anche $SOL_{i+1} \subseteq SOL^*$ allora abbiamo fatto. Altrimenti $SOL_{i+1} \not\subseteq SOL^*$. Sia $j \notin SOL^*$ l'indice aggiunto a SOL_i dall'algoritmo e j' l'indice minimo in $SOL^* - SOL_i$. Definiamo $\overline{SOL} = (SOL^* \cup \{j\}) - \{j'\}$. Poichè $|\overline{SOL}| = |SOL^*|$ resta solo da dimostrare che $\overline{SOL}$, al pari di SOL^*, è una partizione di V in sottovettori ciascuno dei quali ha elementi la cui somma non supera b. Sia $[a, j]$ il sottovettore che termina in j individuato da SOL_{i+1}, per ipotesi induttiva e poichè j' è il minimo in $SOL^* - SOL_i$, nella partizione individuata da SOL^* è presente il sottovettore $[a, j']$. L'algoritmo ha inserito j in SOL perchè la somma dei valori del sottovettore $[a, j]$ più il valore di $V[j + 1]$ supera b. Questo implica che $j' < j$ (in caso contrario il sottovettore $[a, j']$ presente nella partizione individuata da SOL^* avrebbe somma degli elementi superiore a b). In SOL^* la sostituzione di j' con j ha variato solo due sottovettori della partizione: quello che terminava con

j' e quello che iniziava con $j'+1$, il primo è stato sostituito dal sottovettore $[a, j]$ la somma dei cui valori, per come funziona il nostro algoritmo, non supera b mentre il secondo, avendo sostituito j' con j dove $j' < j$ è stato sostituito da un sottovettore la somma dei cui valori è inferiore a quella del sottovettore precedente e quindi non può superare b.

Per la scelta degli indici fatta dall'algoritmo i sottovettori della partizione individuata da SOL hanno tutti somma degli elementi che non supera il valore b, inoltre abbiamo appena dimostrato che $SOL \subseteq SOL^*$, dalla minimalità della partizione individuata da SOL^* segue anche la minimalità della partizione individuata da SOL.

2. L'algoritmo PART2 non risolve il problema. Infatti, sull'istanza $V = [2, 1, 1, 1, 1, 2]$ e $b = 4$, l'algoritmo produrrebbe l'output $\{1, 5, 6\}$, mentre una partizione ottima è $\{3, 6\}$.