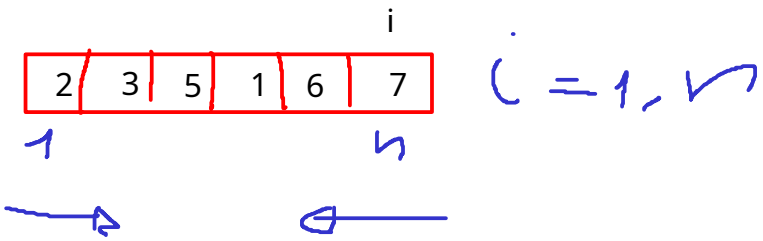
Esercizio 3 [11 punti] I tuoi amici Alice e Bob ti hanno invitato a cena a casa loro e avete ordinato una teglia di pizza con gusti misti. La pizza arriva in un cartone rettangolare che contiene n pezzi di pizza già tagliati allineati in una lunga striscia. Non tutti i gusti ti piacciono allo stesso modo. Se mangi il pezzo i-esimo (immagina i pezzi numerati da sinistra a destra) godi g_i. Ora, quando prendi un pezzo di pizza, per questioni di praticità, puoi scegliere solo uno dei due pezzi esterni (quelli interni si porterebbero via troppa mozzarella dai pezzi adiacenti e i tuoi amici non sarebbero contenti). Tu sei ospite e sei invitato per primo a scegliere il primo pezzo da mangiare. Una volta scelto il pezzo, Alice e Bob si mangeranno i due più esterni fra quelli che restano, e poi toccherà ancora te scegliere. Se per esempio tu decidi di mangiare il pezzo numero 1 (quello più a sinistra), Alice e Bob mangeranno il numero 2 e il numero n. Mentre se scegli il pezzo più a destra, loro mangeranno il numero 1 e il numero $n-1$. E così via.

Progetta un algoritmo di programmazione dinamica che calcola il massimo godimento ottenibile, dove il godimento che ottieni da un sottoinsieme di pezzi $S \subseteq \{1, 2, \ldots, n\}$ è definito come

$$g(S) = \sum_{i \in S} g_i.$$

Si discuta la complessità temporale dell'algoritmo proposto.



OPT(i, j): il godimento massimo ottenibile da una sequenze di pizze da i - j

1. se mangio il pezzo i-esimo pezzo -> ottengo gi, + OPT(i + 2, j - 1)
2. se mangio il pezzo j-esimo pezzo -> ottengo gj + OPT(i + 1, j - 2)

3. Caso base: OPT(i, j) = gi if i = j

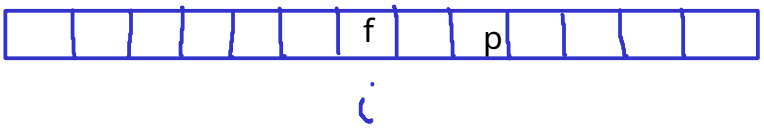
Formula di Bellman:

OPT(i, j) = gi if i = j
OPT(i, j) = max{gi + OPT(i + 2, j - 1), gj + OPT(i + 1, j - 2)} i != j

Esercizio 3 [13 punti] Il signor Marche è tornato da un lungo periodo di vacanza fra Roma e Firenze e sta per affrontare un lungo periodo di lavoro che durerà n giorni. Il suo è un lavoro a cottimo e i soldi che riesce a portarsi a casa dipendono da *quanto*, *quando* e *come* lavorerà. Lui sa lavorare in due modi: *forte* e *piano*. Se nel giorno i lavora forte guadagnerà f_i euro, mentre se lavora piano guadagnerà p_i euro. Chiaramente $f_i \geq p_i$. A rendere le cose difficili ci sono dei vincoli di stanchezza. Dopo aver lavorato per un giorno in modo forte, il signor Marche ha bisogno di riposarsi (e quindi di non lavorare) per i successivi due giorni. Inoltre, in ogni caso lui non riesce a lavorare due giorni di seguito. Come potete immaginare i giorni in cui non lavora non guadagna niente.

Progettare un algoritmo di programmazione dinamica che calcoli un piano di lavoro per il signor Marche che gli faccia guadagnare il più possibile.

pi skippa 1 giorno
fi skippa 2 giorni
fi >= pi



OPT(i): massimo guadagno a partire dal giorno i ... n

GOAL: OPT(1)

Formula di Bellman

OPT(i) = 0 per i > n
OPT(i) = max{fi + OPT(i + 3), pi + OPT(i + 2), OPT(i + 1)}

- Se lavoro f: per i prossimi 2 giorni sto fermo e guadagno fi
OPT(i) = fi + OPT(i + 3)
- Se lavoro p: il prossimo giorno sto fermo e guadagno pi
OPT(i) = pi + OPT(i + 2)

OPT(i) = 0 per i > n