Esercizio 3 [13 punti]
Sia data una scacchiera rettangolare $1 \times n$ di n celle. Su ogni cella i è possibile posizionare una pedina nera ottenendo n_i punti, oppure una pedina bianca ottenendo b_i punti con $b_i \geq n_i$. Un piazzamento di pedine si dice *indipendente* se per ogni coppia di celle adiacenti almeno una delle due è vuota. Progettare un algoritmo di programmazione dinamica che trova un piazzamento indipendente di punteggio massimo, sapendo di avere a disposizione n pedine nere e solo una pedina bianca.

Esercizio 3 [13 punti]
Vi chiedono di giocare al seguente gioco. Potete piazzare delle pedine su una scacchiera $1 \times n$. Avete a disposizione pedine dei seguenti tre colori, elencati dal più *chiaro* al più *scuro*: bianco, grigio e nero. Piazzare una pedina nella posizione i vi fa guadagnare b_i se la pedina è bianca, g_i se la pedina è grigia, n_i se è nera. Avete i seguenti vincoli:

- se piazzate due pedine adiacenti, diciamo in posizione i e $i+1$, allora la pedina in posizione i deve essere più chiara di quella in posizione $i+1$;
- se in una certa posizione i non avete piazzato pedine, allora nella posizione $i+1$ potete piazzare una pedina di un qualsiasi colore;
- la prima pedina da sinistra che piazzate deve essere bianca;

Progettare un algoritmo di programmazione dinamica che calcola il massimo punteggio ottenibile.

Esercizio 3 [13 punti]
Su una scacchiera $2 \times n$ potete piazzare esattamente n pedine, una per ogni colonna. Per ogni pedina potete scegliere il colore, bianco o nero. Piazzare una pedina nella posizione (i, j) vi fa guadagnare $b_{i,j}$ se la pedina è bianca, e $n_{i,j}$ se è nera. Avete il seguente vincolo: per ogni $j > 1$, la pedina che piazzate sulla colonna j deve avere lo stesso colore oppure trovarsi nella stessa riga della pedina che avete piazzato nella colonna $j-1$.

Progettare un algoritmo di programmazione dinamica che calcola il massimo punteggio ottenibile.

Esercizio 3 [13 punti]
State progettando una vacanza intergalattica con la vostra astronave e dovete fare n viaggi. La vostra astronave può volare in quattro diverse modalità (A, B, C e D). Effettuare il viaggio i-esimo nella modalità $x \in \{A, B, C, D\}$ ha un costo $c_{i,x}$. Una volta scelta una modalità per un viaggio non è possibile cambiarla in volo. Però alla fine di ogni viaggio potete decidere di riconfigurare l'astronave e cambiare modalità. Il cambio di modalità ha un costo di riassetto che conoscete. In particolare, per ogni $x, y \in \{A, B, C, D\}$, sia $\delta_{x,y}$ il costo per passare dalla modalità x alla modalità y. Si assuma che $\delta_{x,x} = 0$ per ogni x. Per ragioni tecniche dovete partire per il primo viaggio con l'astronave in modalità A e arrivare a destinazione (dopo gli n viaggi) in modalità C o D.

Progettare un algoritmo di programmazione dinamica che calcola il costo di una vacanza di costo minimo.

Esercizio 3 [13 punti]
Nel corridoio del museo di Tor VerLouvre ci sono n vani, ognuno dei quali ospita un quadro. Si vogliono istallare delle telecamere per controllare gli n quadri del corridoio. In corrispondenza di ogni vano i, è possibile istallare una telecamera e regolarne il raggio di azione $r \in \{0, \ldots, n-1\}$. Una telecamera istallata nel vano i con raggio di azione r è in grado di controllare i quadri presenti dal vano i al vano $i+r$ e costa (in termini di consumo energetico) $c_{i,r}$. Si assuma che per ogni i, il costo $c_{i,r}$ è crescente al crescere del raggio r.

Progettare un algoritmo di programmazione dinamica che, presa in input la matrice dei costi $c_{i,r}$, calcola il costo minimo necessario per controllare gli n quadri del corridoio.

Esercizio 3 [13 punti]
State progettando una traversata intergalattica con la vostra astronave e dovete fare n viaggi. La vostra astronave può volare in due diverse modalità (A e B). Effettuare il viaggio i-esimo nella modalità $x \in \{A, B\}$ ha un costo $c_{i,x}$. La vostra astronave si trova inizialmente in modalità A. Una volta scelta una modalità per un viaggio non è possibile cambiarla in volo. Però all'inizio di ogni viaggio potete decidere di riconfigurare l'astronave e cambiare modalità (anche all'inizio del primo viaggio). Il cambio di modalità non costa niente, ma avete pezzi di ricambio che vi consentono di cambiare modalità al più tre volte. Potete arrivare a destinazione in una qualsiasi modalità.

Progettare un algoritmo di programmazione dinamica che calcola il costo di una traversata di costo minimo.

Esercizi d'esame 2016/2017

Esercizio 3 [13 punti]
Sia data una scacchiera rettangolare $1 \times n$ di n celle. Su ogni cella i è possibile posizionare una pedina nera ottenendo n_i punti, oppure una pedina bianca ottenendo b_i punti con $b_i \geq n_i$. Un piazzamento di pedine si dice *indipendente* se per ogni coppia di celle adiacenti almeno una delle due è vuota. Progettare un algoritmo di programmazione dinamica che trova un piazzamento indipendente di punteggio massimo, sapendo di avere a disposizione n pedine nere e solo una pedina bianca.

Esercizio 3 [13 punti]
Si vuole illuminare una pista ciclabile di n tratte. In ogni tratta è possibile, volendo, installare un lampione. Se installato sulla tratta i, un lampione è in grado di illuminare la tratta i e le due tratte immediatamente precedenti e immediatamente successive i (ovvero le tratte $i-2$ e $i-1$, se esistono, e le tratte $i+1$ e $i+2$, se esistono). Il costo di installazione, però, non è uniforme. Più precisamente, installare un lampione nella tratta i costa c_i. Progettare un algoritmo di programmazione dinamica che trova il costo minimo con cui è possibile illuminare l'intera pista ciclabile, ovvero il costo di una soluzione migliore possibile in cui ogni tratta è illuminata da almeno un lampione.

Esercizio 3 [13 punti]
Il prof. Gualà è preoccupato per la sua reputazione: sembra ormai ufficiale che la prof.ssa Di Ianni bocci agli esami più di quanto faccia lui. Ha quindi deciso, usando le sue competenze algoritmiche, di prendere dei provvedimenti per i prossimi n appelli che verranno. Sa che ad ogni appello può preparare compiti scritti di diversa difficoltà: facile, medio, difficile. Per ogni appello $i = 1, \ldots, n$, il prof. Gualà stima che il numero di bocciati saranno f_i, m_i o d_i se assegnerà uno scritto rispettivamente facile, medio, o difficile, con $f_i \leq m_i \leq d_i$. Deve però stare attento, perché se esagerasse i rappresentanti degli studenti lo dipingerebbero come un tiranno insensibile e ingiusto e quindi la sua reputazione crollerebbe. Deve quindi attenersi ai seguenti vincoli: se in un certo appello assegna un compito difficile, allora nei successivi tre (se ci sono)[1] lo scritto deve essere facile. Uno scritto medio deve sempre essere seguito (se c'è un appello successivo) da uno scritto facile.

Progettate un algoritmo di programmazione dinamica che calcoli la strategia migliore per il prof. Gualà, ovvero il modo migliore di assegnare gli scritti negli appelli al fine di massimizzare il numero di bocciati.

Esercizio 3 [13 punti]
Nella versione Beta di Pokemon Go la mappa è una sola strada di $n + 1$ tratte. Tu sei nella posizione 0 e ad ogni istante di tempo sei costretto a spostarti a destra di una posizione, dalla posizione corrente i alla posizione $i + 1$. In ogni posizione $i \geq 1$ ti si presenta davanti un pokemon. Tu puoi fare una sola delle seguenti due mosse:

- raccogliere le k_i pokemon ball che sono nella posizione i;
- lanciare al pokemon delle pokemon ball al fine di catturarlo. Il numero di ball che ti permettono di catturare il pokemon nella posizione i è p_i e tu, chiaramente, ne devi precedentemente aver raccolte a sufficienza.

All'inizio, quando sei nella posizione 0, non possiedi pokemon ball. Chiaramente $\Delta = \sum_{i=1}^{n} k_i$ è il numero massimo di pokemon ball che puoi raccogliere nell'intero percorso. Progettare un algoritmo di programmazione dinamica che calcoli il numero massimo di pokemon catturabili. L'algoritmo deve avere complessità polinomiale in n e Δ.

Esercizi d'esame 2015/2016

Esercizio 3 [13 punti]
Un drone equipaggiato con una batteria a energia solare si muove su un territorio descritto da una matrice M con n righe e m colonne. Deve andare dal punto in alto a sinistra, di cordinate $(1, 1)$, al punto in basso a destra, di cordinate (n, m). Soffia un forte vento che impedisce al drone di spostarsi in tutte le direzioni. In particolare, il drone può spostarsi ogni volta solo di una cella a destra o di una cella in basso rispetto alla posizione in cui si trova, operazione che gli fa consumare una unità di energia della batteria. La batteria ha un capacità massima di Δ unità, all'inizio completamente carica. Ovviamente il drone non può mouversi se il livello della sua batteria è a 0. Esso, però, è equipaggiato di pannelli ricettivi che gli permettono di ricaricare la batteria qualora passasse in punti in cui i raggi solari sono particolarmente forti. Più precisamente, per ogni posizione (i, j), è noto il numero $M[i, j] \geq 0$ di unità di energia che si ottengono quando si passa nella posizione (i, j). La batteria non può accumulare più di Δ unità di energia, ma non si danneggia se sottoposta a raggi solari quando è già al massimo della carica. Progettare un algoritmo di programmazione dinamica che calcoli quale è il massimo livello di batteria con cui il drone può arrivare a destinazione.

Esercizio 3 [13 punti]
In una strada del centro ci sono n stalli sui cui potete collocare delle attività commerciali. Potete scegliere attività di due tipi, di tipo A e di tipo B. Intuitivamente, l'attività di tipo B vi permette di guadagnare di più ma ha dei vincoli legislativi più stringenti. In particolare, se decidete di collocare un'attività di tipo B nello stallo i, dovete lasciare liberi gli stalli $i-1$ e $i+1$, su cui quindi non potete collocare nessuna attività, né di tipo A né di tipo B. L'attività di tipo A, invece, essenzialmente non ha vincoli (se non quello chiaramente di non poter essere adiacente ad un'attività di tipo B). Il vostro obiettivo è ovviamente quello di fare più soldi possibile. A tale fine avete fatto una ricerca di mercato e sapete che collocare un'attività di tipo A nello stallo i vi farebbe guadagnare a_i, mentre una di tipo B vi farebbe guadagnare b_i (con $b_i \geq a_i$). Progettate un algoritmo di programmazione dinamica che calcoli il guadagno massimo che potete ottenere dalla strada del centro.

Esercizio 3 [13 punti]
Si vogliono posizionare delle pedine su una scacchiera rettangolare $2 \times n$. Ad ogni cella (i, j) è assegnato un valore positivo $v_{i,j}$, $i \in \{1, 2\}, j \in \{1, \ldots, n\}$, che indica il guadagno che si ottiene se si posiziona una pedina sulla cella (i, j). Due pedine non devono mai essere adiacenti in senso verticale o orizzontale (ma possono esserlo diagonalmente). Si progetti un algoritmo di programmazione dinamica che calcoli il guadagno massimo ottenibile posizionando nel miglior modo possibile le pedine.

Esercizio 3 [13 punti]
Il prof. Gualà è in ufficio e deve necessariamente rimanere sveglio per le prossime n ore (indicizzate da 1 ad n). Per non addormentarsi, all'inizio di ogni ora, può scegliere di prendere un caffè. Esistono due tipi di caffè: (i) il caffè *ristretto* costa r euro e tiene sveglio per 2 ore (quella corrente e quella successiva), (ii) il caffè *lungo* costa ℓ euro, con $\ell > r$ e tiene svegli per 4 ore consecutive (quella corrente e le successive tre). Inoltre, in alcune ore, il prof. Gualà è impegnato in riunioni importanti (e noiose) e non può procurarsi il caffè. In particolare, poniamo $b_i = 1$ se è prevista una riunione nell'i-esima ora, e $b_i = 0$ altrimenti. Progettare un algoritmo di programmazione dinamica che, presi in input il numero di ore n, i costi dei caffè r ed ℓ ed i valori di b_i, calcoli quando e quali caffè deve comprare il prof. Gualà in modo da spendere il meno possibile senza addormentarsi.

Esercizi d'esame 2014/2015

Esercizio 3 [13 punti]
Il prof. Gualà è in ufficio e deve necessariamente rimanere sveglio per le prossime n ore (indicizzate da 1 ad n). Per non addormentarsi, all'inizio di ogni ora, può scegliere di prendere un caffè. Esistono due tipi di caffè: (i) il caffè *ristretto* costa r euro e tiene sveglio per 2 ore (quella corrente e quella successiva), (ii) il caffè *lungo* costa ℓ euro, con $\ell > r$ e tiene svegli per 4 ore consecutive (quella corrente e le successive tre). Inoltre, in alcune ore, il prof. Gualà è impegnato in riunioni importanti (e noiose) e non può procurarsi il caffè. In particolare, poniamo $b_i = 1$ se è prevista una riunione nell'i-esima ora, e $b_i = 0$ altrimenti. Progettare un algoritmo di programmazione dinamica che, presi in input il numero di ore n, i costi dei caffè r ed ℓ ed i valori di b_i, calcoli quando e quali caffè deve comprare il prof. Gualà in modo da spendere il meno possibile senza addormentarsi.

Esercizio 3 [13 punti] Il signor Valter Bianchi, fuorviato da una recente serie TV americana, ha deciso di fare soldi vendendo cristalli non proprio legali. Ha a disposizione una quantità iniziale Δ di cristalli, e conosce n potenziali compratori. Il compratore i vuole esattamente x_i cristalli ed è disposto a pagare p_i. Progettare un algoritmo di programmazione dinamica che calcoli il massimo guadagno ottenibile dal signore Valter Bianchi. L'algoritmo deve avere complessità temporale $O(n\Delta)$.

Esercizio 3 [13 punti]
Il signor Valter Bianchi, dopo aver fatto un bel gruzzoletto vendendo cristalli non proprio legali, ha deciso di diversificare la sua attività e si è messo a vendere della roba – anch'essa non proprio legale – che per comodità chiameremo stecca di cioccolata. La stecca di cioccolata può essere venduta tutta intera o può essere spezzata in segmenti più piccoli da vendere separatamente. La lunghezza della stecca di cioccolata è di L centimetri, con L intero. Si assuma che nello spezzare la stecca la lunghezza dei pezzi ottenuti (in centimetri) debba essere ancora un numero intero. Per esempio un pezzo lungo 3 centimetri può essere venduto così, spezzato in tre pezzi da 1 centimetro o in due pezzi: uno da 2 centimetri e l'altro da 1 centimetro (mentre non si possono fare due pezzi da mezzo centimetro e due centimetri e mezzo). Il guadagno che il signor Valter Bianchi riesce a fare se vende un pezzo lungo t centimetri è $G(t)$, $t = 1, 2, \ldots, L$. Progettare un algoritmo di programmazione dinamica che aiuti il signor Valter Bianchi a guadagnare il più possibile. La complessità temporale dell'algoritmo deve essere polinomiale in L.

Esercizio 3 [13 punti]
Sonic si trova su una piattaforma composta da n caselle e deve raccogliere più anelli possibile. Le caselle, da sinistra a destra, sono numerate da 1 a n. Lui si trova nella casella 1 e il traguardo che deve superare è alla casella n. Ogni casella i ha un certo numero t_i di anelli a terra e un certo numero a_i di anelli in aria. Sonic ogni volta può fare due mosse: *spostarsi* a destra o *saltare* a destra. Se lui si trova nella casella i e si sposta a destra, allora finisce nella casella $i+1$ e raccoglie gli anelli a terra che si trovano in tale casella. Se invece decide si saltare a destra, allora si sposta nella casella $i+3$, raccoglie gli anelli che si trovano a terra nella casella $i+3$ e quelli che si trovano in aria nelle caselle $i+1$ e $i+2$. Progettare un algoritmo di programmazione dinamica che calcola in numero massimo di anelli che Sonic può raccogliere prima di tagliare il traguardo. Si assuma che non è necessario finire a terra nella casella n, ma è possibile tagliare il traguardo anche in volo.

Esercizio 3 [13 punti] *Il signor Marche va a Venezia*
Il signor Marche si è recato in vacanza a Venezia e ha scoperto, con suo disappunto, quanto è cara la famosa città veneta. Ora si trova nella condizione di dover attraversare un canale in gondola e, guardando i prezzi sui volantini, gli è preso un colpo. Questo canale, che può essere percorso in un'unica direzione, passa per n diversi porticcioli, che per comodità numereremo da 1 a n. In ogni porticciolo è possibile imbarcarsi su una gondola e scendere lungo il canale verso porti successivi. Il signor Marche si trova nel porto 1 e deve raggiungere il porto n. Il costo di imbarco dipende dal porto di partenza e da quello di arrivo. Più precisamente, imbarcarsi nel porto i e scendere nel porto j costa $c(i, j)$ $(i > j)$. Progettate un algoritmo di programmazione dinamica che aiuti il signor Marche a raggiungere l'ultimo porto spendendo il meno possibile e appagando così, almeno in parte, il suo proverbiale attaccamento ai soldi. L'algoritmo, che deve calcolare la miglior soluzione prima del tramonto, deve avere complessità polinomiale nella dimensione dell'istanza.