Laboratorio di Algoritmi

Progetto "Slideshow" (gennaio 2026)

Nota: La scadenza del progetto è fissata per lunedì 19 gennaio **compreso**.

Nota: Si consiglia di consultare sulla pagina web il documento che riporta le avvertenze utili per lo svolgimento del progetto. Si consiglia anche di verificare di tanto in tanto gli aggiornamenti a questo documento, che potranno riportare risposte ai dubbi degli studenti e correzioni di eventuali errori.

Nota: Questa versione del documento è stata aggiornata il 5 gennaio 2026 per specificare che il numero di immagini verticali sarà sempre pari.

Il problema Si ha una collezione di immagini digitali, con la quale si vuole creare una presentazione da proiettare o mettere in rete sequenzialmente, ovvero uno *slideshow*. Ogni foto ha un contenuto descritto da un elenco di temi, ovvero *tag*. Alcune immagini sono orientate orizzontalmente, altre verticalmente. La proiezione mostrerà le immagini orizzontali una alla volta, quelle verticali a coppie affiancate fra loro. Per semplicità ipotizzeremo che il numero di immagini verticali sia sempre pari, in modo da poterle sempre accoppiare.

Il problema richiede anzi tutto una serie di semplici elaborazioni e stampe riguardanti i dati. In particolare, si tratta di stampare informazioni sul numero e l'orientamento delle foto, sulla distribuzione dei temi fra le foto, sull'insieme dei temi. Queste informazioni sono descritte in maggiore dettaglio nel seguito.

Quindi si deve affrontare il problema di costruire una proiezione che abbia il massimo interesse possibile per gli spettatori. L'interesse di una proiezione è legato al modo in cui si affiancano e susseguono i temi espressi dalle immagini. Più precisamente, definiamo T_s l'insieme dei temi delle immagini contenute in una schermata s: se s consiste in una semplice immagine orizzontale, T_s coincide con l'insieme dei temi dell'immagine stessa; se s consiste in due immagini verticali accostate, T_s coincide con l'unione dei due relativi insiemi di temi. Si suppone che l'interesse di una proiezione cresca combinando *conferma* e *sorpresa*: la conferma è legata al fatto di vedere in successione stretta immagini con gli stessi temi, mentre la sorpresa è legata al fatto di vedere immagini con temi diversi. In base a questa teoria psicologica, definiamo un *indice di interesse* per ogni transizione diretta da una schermata s a una schermata s', pari a:

$$w_{ss'} = \min\left(\left|T_s \cap T_{s'}\right|, \left|T_s \setminus T_{s'}\right|, \left|T_{s'} \setminus T_s\right|\right)$$

vale a dire la cardinalità minima fra l'intersezione, la differenza diretta e la differenza inversa degli insiemi stessi. Infatti, questo numero tende ad essere alto quando due schermate consecutive hanno molti temi in comune, ma anche quando molti temi nuovi compaiono nella seconda e infine quando molti temi della prima immagine scompaiono passando alla seconda. Definiamo poi l'interesse complessivo di una proiezione come la somma degli indici di interesse per tutte le transizioni (coppie di schermate consecutive) che hanno luogo scorrendo la galleria.

Si tratta ora di decidere come accoppiare le immagini verticali in schermate e in che ordine disporre le schermate per ottenere la proiezione di massimo interesse per gli spettatori. Leschermate costituite da singole immagini orizzontali e quelle costituite da coppie di immagini verticali si possono susseguire in qualsiasi ordine. Questo problema di ottimizzazione ha chiaramente un numero finito di soluzioni, ma l'algoritmo esaustivo che le analizza tutte non è pratico. È possibile dimostrare, sulla base di equivalenze con problemi noti in letteratura, che non sono noti algoritmi esatti di complessità polinomiale per il problema. È quindi giustificato far

ricorso a un algoritmo euristico per risolvere il problema. L'algoritmo da realizzare decompone il problema in due parti:

1. individuazione dell'accoppiamento delle immagini verticali in schermate;
2. individuazione della sequenza delle schermate.

Questo contrasta con il fatto che i due aspetti interagiscono in modo non banale: l'accoppiamento delle immagini verticali influisce sulla qualità del sequenziamento, ma questo a sua volta modifica la qualità dei possibili accoppiamenti. D'altra parte, in questo caso l'efficienza richiede di sacrificare l'efficacia.

Per il sottoproblema di accoppiamento, esiste un algoritmo esatto polinomiale, ma questo eccede i limiti del corso, per cui applicheremo l'algoritmo *greedy* che usa come insieme base quello delle immagini verticali e come collezione degli indipendenti quella formata da tutti i possibili sottoinsiemi di coppie disgiunte di immagini verticali. Sono, cioè, vietati i sottoinsiemi che contengono due coppie con una foto in comune. Siccome nel sottoproblema di sequenziamente vogliamo favorire la presenza di schermate consecutive con molti temi uguali oppure con molti temi diversi, nel sottoproblema di accoppiamento cercheremo di costruire schermate molto ricche di temi. Quindi, l'algoritmo *greedy* sceglierà di volta in volta la coppia di immagini verticali lecita (cioè non intersecante coppie già scelte in precedenza) che abbia il massimo numero totale di temi. In caso di uguaglianza, sceglierà la coppia con il minimo numero di temi comuni alle due foto. In caso di ulteriore parità, sceglierà la coppia le cui foto compaiono per prime nei dati (considerando nell'ordine la foto di sinistra e poi quella di destra nella schermata)[1].

Risolveremo con un algoritmo *greedy* euristico anche il sottoproblema di sequenziamento. L'algoritmo parte dalla schermata che contiene la prima foto orizzontale fornita nei dati, e sceglie ad ogni passo la schermata s' per la quale la transizione dalla schermata corrente s ha il massimo indice di interesse $w_{ss'}$. In caso di parità, massimizza prima la cardinalità dell'intersezione $T_s \cap T_{s'}$, poi quella della differenza diretta $T_s \setminus T_{s'}$, poi quella della differenza inversa $T_{s'} \setminus T_s$. Infine, in caso di ulteriore parità, sceglie la schermata s' la cui prima (o unica) foto compare per prima nei dati. Giunti all'ultima schermata, compie la seguente riottimizzazione secondaria: siccome la schermata scelta convenzionalmente come la prima potrebbe non essere la migliore, l'algoritmo crea una proiezione ciclica tornando dall'ultima schermata alla prima, individua la transizione di interesse minimo lungo l'intera sequenza ciclica (applicando i criteri ausiliari sopra elencati in caso di parità) e la cancella. Ne risulta una proiezione che parte con la schermata finale della transizione cancellata e si conclude con la sua schermata iniziale.

Il progetto Il programma da realizzare carica i dati da un file di testo il cui nome va fornito nella linea di comando. Il file riporta nella prima riga il numero di foto. Seguono altrettante righe, ciascuna delle quali riporta l'orientamento della foto corrispondente (`O` o `V`, cioè orizzontale o verticale), il numero di temi e l'elenco dei temi stessi. Ciascuno di questi è, per semplicità, una singola parola composta di al più 10 caratteri alfanumerici (lettere o cifre)[2]. Per esempio:

[1] Per essere più chiari, anche se le foto nel file dei dati non sono numerate, le tratteremo come se avessero un indice progressivo crescente e useremo questo indice per prendere decisioni quando i criteri indicati nel testo non sono sufficienti a distinguere tra scelte alternative.

[2] Questa indicazione serve solo a dimensionare le strutture per contenere i temi. Non va intesa come un'istigazione a ignorare la lunghezza delle stringhe nelle analisi di complessità spaziale e

```
4
O 3 sole mare2 tramonto
V 2 giardino1 gatto
V 2 giardino1 rosa
O 2 lago3 tramonto
```

indica che la galleria conterrà 4 immagini. La prima ha i tre temi `sole`, `mare2` e `tramonto`. Seguono due immagini verticali, che hanno entrambe due temi, rispettivamente `giardino1` e `gatto` per la prima, `giardino1` e `rosa` per la seconda. Chiude l'elenco una immagine verticale con i due temi `lago3` e `tramonto`.

Il programma deve stampare a video su una riga il numero n_o delle foto orizzontali, seguito dalle parole chiave `foto orizzontali`, e sulla riga seguente il numero n_v delle foto verticali, seguito dalle parole chiave `foto verticali`. Nel caso dell'esempio:

```
2 foto orizzontali
2 foto verticali
```

Quindi, deve stampare una riga per ciascun numero possibile di temi, calando dal massimo al minimo, riportando quante foto hanno esattamente tale numero di temi, e saltando i valori a cui non corrisponde alcuna foto, nel formato seguente:

```
3 tag 1 foto
2 tag 3 foto
```

Per concludere l'analisi sui dati, il programma deve stampare due volte l'elenco dei temi, preceduto dal loro numero. La prima volta li stamperà in ordine alfabetico, uno per riga; la seconda volta per numero decrescente di foto nelle quali compaiono (e, a parità, in ordine alfabetico), sempre uno per riga, facendo seguire al temi il numero delle sue occorrenze. Nel caso dell'esempio, la prima stampa sarà:

```
7
gatto
giardino1
lago3
mare2
rosa
sole
tramonto
```

mentre la seconda sarà

```
7
giardino1 2
tramonto 2
gatto 1
lago3 1
mare2 1
rosa 1
sole 1
```

Rimane da stampare la soluzione del problema di ottimizzazione. Il programma deve stampare una prima riga contenente l'indice totale di interesse della galleria, seguita da una seconda riga che riporta il numero di schermate. A questa devono seguire le schermate stesse, una per riga: per ciascuna si indicherà l'indice numerico

temporale, né tanto meno a imporre un limite massimo al numero di temi per ogni foto o al numero complessivo di temi. Allo stesso modo, la lettura dei file di esempio non deve istigare ipotesi sui nomi dei temi (come il fatto che comincino tutti per `t` o amenità del genere).

della foto orizzontale o i due indici numerici delle foto verticali che compaiono nella schermata. Questi indici sono i numeri d'ordine con cui le foto si presentano nel file dei dati (da 1 fino al numero totale di foto), e va stampato sempre prima il minore, poi il maggiore. Nel caso dell'esempio, il primo algoritmo *greedy* è banale, perché ci sono solo due foto verticali, che vanno necessariamente accoppiate. La schermata risultante ha i tre temi `giardino1`, `gatto` e `rosa`. Il secondo algoritmo *greedy* parte dalla schermata corrispondente alla prima foto orizzontale. Deve scegliere fra due transizioni:

1. da {`sole, mare2, tramonto`} a {`giardino1, gatto, rosa`}, con intersezione vuota, e differenze coincidenti con i due insiemi T_s e $T_{s'}$, per cui $w_{ss'} = \min(0, 3, 3) = 0$;
2. da {`sole, mare2, tramonto`} a { `lago3, tramonto`}, con intersezione {`tramonto`}, differenza diretta {`sole, mare2`} e differenza inversa {`lago3`}, per cui $w_{ss'} = \min(1, 2, 1) = 1$.

Sceglie quindi la seconda. Dopo di che, necessariamente, si passa alla schermata mancante, con una transizione da {`lago3, tramonto`} a {`giardino1, gatto, rosa`}, con $w_{ss'} = \min(0, 2, 3) = 0$. Per concludere, cancella la transizione di indice minimo. Ve ne sono due con indice nullo:

1. da {`lago3, tramonto`} a {`giardino1, gatto, rosa`}
2. da {`giardino1, gatto, rosa`} a {`sole, mare2, tramonto`}

Entrambe hanno intersezione vuota e differenze coincidenti con i due insiemi, per cui vince la prima, che ha la differenza diretta di cardinalità 2 anziché 3. Rotta la sequenza ciclica in tale transizione, la nuova sequenza rimane costituita dalla schermata con le due immagini verticali (che avevano indice 2 e 3 nel file dei dati), seguita dalla prima immagine orizzontale (di indice 1) e poi dalla seconda (di indice 4). L'indice complessivo di interesse è pari a 1.

```
1
3
2 3
1
4
```

Chiarimenti