Laboratorio di Algoritmi e Strutture Dati

Docente: V. Lonati

Progetto "Piastrelle digitali"[1]

valido per gli appelli di giugno, luglio e settembre 2024

Indice

1 Organizzazione degli appelli e modalità di consegna

La realizzazione del progetto è una prova d'esame da svolgersi **individualmente**. I progetti giudicati frutto di **copiatura** saranno **estromessi** d'ufficio dalla valutazione.

Si richiede allo studente di effettuare un **adeguato collaudo** del proprio progetto su numerosi esempi diversi per verificarne la correttezza.

La versione aggiornata del progetto è pubblicata nella pagina del corso:
`https://myariel.unimi.it/course/view.php?id=1221`

Si consiglia di consultare periodicamente questo sito per eventuali correzioni e/o precisazioni relative al testo del progetto. Per ogni ulteriore chiarimento potete chiedere un appuntamento scrivendo una mail all'indirizzo `lonati@di.unimi.it` .

La presente traccia è valida per gli appelli di giugno, luglio e settembre 2024.

Il progetto svolto va consegnato tramite il sistema di upload (`https://upload.di.unimi.it`) del Dipartimento entro una delle seguenti scadenze:

1. Prima scadenza: 10 giugno 2024. Periodo per le discussioni: 17-28 giugno 2024
2. Seconda scadenza: 10 luglio 2024. Periodo per le discussioni: 22-26 luglio 2024
3. Terza scadenza: 8 settembre 2024. Periodo per le discussioni: 16-25 settembre 2024

Le specifiche variano leggermente a seconda della scadenza, secondo quanto indicato nelle sezioni successive.

[1] Ultima modifica 13 maggio 2024

Occorre consegnare un programma go e una relazione. Più precisamente:

1. Il programma go (file sorgenti) deve essere preparato secondo le specifiche descritte nelle sezioni successive. Se il programma è formato da un solo file, il nome del file contenente il `main` deve essere nel formato `123456_cognome_nome.go`, dove `123456` è la matricola. Nel caso in cui il programma sia organizzato su più file, la relazione deve iniziare con un'indicazione di come è organizzato il programma e di come compilarlo.

2. La relazione deve essere in formato `.pdf`, e avere lunghezza compresa tra 3 e 10 pagine. La relazione deve descrivere in modo sintetico il programma, spiegando in che modo questo affronta il problema descritto nella traccia. Illustrate sia le scelte di progetto in relazione al problema (ad esempio; come è stato modellato il problema, quali strutture di dati sono state usate e perché) sia quelle implementative in relazione al programma (ad esempio: dettagli su come sono stati implementati gli algoritmi o su come sono state implementate le strutture dati scelte in fase di progetto).

   La relazione deve inoltre contenere una rassegna di esempi esempi (diversi da quelli già proposti nella traccia).

3. Opzionalmente possono essere consegnati anche file di test in formato `go`, oppure in formato `.txt` (coppie di input/output)

Tutti i file devono essere contenuti in un unico archivio `.zip` con nome `123456_cognome_nome.zip`. Tutti i file nell'archivio, compresa la relazione, devono riportare nome, cognome e matricola dell'autore.

È possibile consegnare una sola volta; se un progetto presentasse delle mancanze o venisse valutato non pienamente sufficiente, potranno essere richieste integrazioni o revisioni del lavoro svolto.

Dopo ciascuna scadenza per la consegna, i progetti consegnati verranno corretti. in fase di correzione si valuterà chi dovrà discutere il progetto (la discussione non è obbligatoria); al termine della correzione verrà pubblicato un avviso con il calendario delle discussioni e l'elenco delle persone convocate per la discussione.

È necessario presentarsi alla discussione con un computer portatile con i file consegnati (oppure richiedere con adeguato anticipo di svolgere la discussione in una aula con computer).

2 Criteri di valutazione dei progetti

Nella valutazione del progetto si terrà conto dei seguenti aspetti: capacità di modellare il problema mediante strutture di dati opportune; capacità di progettare soluzioni algoritmiche efficienti; capacità di implementare in Go le strutture di dati e gli algoritmi scelti, in maniera appropriata rispetto al tipo di elaborazione richiesta; chiarezza espositiva e proprietà di linguaggio nell'illustrare le scelte di modellazione e di implementazione fatte (sia in relazione al problema che in relazione al codice); correttezza formale (sintattica) del codice Go prodotto; funzionamento del programma (correttezza e completezza degli output prodotti dal programma); qualità del codice (codice non ripetuto/ridondante/intricato/oscuro, strutturazione del codice, ecc).

Il progetto verrà estromesso dalla valutazione in uno qualunque dei seguenti casi:

1. i file consegnati non rispettano il formato specificato sopra;

2. il programma non compila;

3. il programma non contiene la definizione dei tipi e delle funzioni indicate nelle Specifiche di implementazione;

4. il programma contiene le funzioni specificate, ma con segnatura diversa.

3 Il problema

Obiettivo del progetto è studiare le configurazioni di insiemi di piastrelle digitali su un piano, e la loro influenza sulle piastrelle circonvicine.

Piastrelle

Il piano è suddiviso in quadrati di lato unitario, ognuno dei quali è occupato da una *piastrella*. In un dato istante, ogni piastrella può essere *accesa* o *spenta*.

Più precisamente, data una coppia di numeri naturali (a, b), chiamiamo Piastrella(a, b) la piastrella di lato unitario che ha come vertici i quattro punti (a, b), $(a, b + 1)$, $(a + 1, b + 1)$, $(a, b + 1)$,

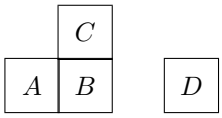
Esempio

Piastrella$(3, 5)$ ha per vertici i punti $(3, 5), (3, 6), (4, 6), (4, 5)$.

Quando è accesa, una piastrella appare colorata. L'insieme dei colori disponibili è l'insieme delle stringhe sull'alfabeto $\{a, b, \ldots, z\}$. Ogni piastrella può essere accesa con intensità differente. L'intensità di una piastrella è indicata con un numero intero (maggiore il numero, maggiore l'intensità).

Piastrelle circonvicine e piste

Diciamo che due piastrelle sono *circonvicine* se hanno in comune almeno un punto (quindi, ogni piastrella è circonvicina a se stessa); nella figura qui sotto le piastrelle A e B sono circonvicine, B e C sono circonvicine, e anche A e C sono circonvicine. Al contrario D non è circonvicina ad alcuna delle altre tre piastrelle.



Una *pista* da A_1 a A_h è una sequenza di piastrelle accese $A_1, \ldots, A_h$ tali che A_i è circonvicina a A_{i+1} per ogni $1 \leq i \leq h - 1$. Una *regione* $\mathcal{R}$ è un insieme di piastrelle accese tali che, per ogni $A, B \in \mathcal{R}$, esiste una pista da A a B contenuta in $\mathcal{R}$.

L'*intensità* di una pista è data dalla somma delle intensità delle piastrelle che la compongono.

Data una posizione (x, y) e una sequenza di direzioni $s = d_1, d_2, \ldots, d_n$, consideriamo la sequenza di piastrelle che parte da e da (x, y) e prosegue spostandosi verso le posizioni circonvicine, in base alla direzioni indicate. Se tutte queste piastrelle sono accese, queste costituiscono la *pista che parte da* (x, y) *e segue* s. Le direzioni possibili sono $\{NO, NN, NE, EE, SE, SS, SO, OO\}$ dove NO indica il nord-ovest, NN indica il nord, e così via.

Esempio

Partendo dalla piastrella A nell'esempio precedente, la sequenza di direzioni EE, NN indica la pista formata dalle piastrelle A, B, C, mentre la sequenza di direzioni NE, SS indica la pista formata dalle

piastrelle A, C, B. Se A, B, C hanno intensità 2, 5 e 4, rispettivamente, allora entrambe le piste hanno intensità 11.

La sequenza di direzioni NN, EE invece non definisce alcuna pista, poiché la piastrella a nord di A è spenta.

Blocchi

Un *blocco* è una regione massimale di piastrelle accese; più precisamente una regione $\mathcal{R}$ è un blocco se, per ogni piastrella accesa $P \notin \mathcal{R}$, $\mathcal{R} \cup \{P\}$ non è una regione.

Il *blocco di appartenza* di Piastrella(x, y) è il blocco che contiene Piastrella (x, y).

Il *perimetro* di un blocco è dato dall'insieme dei lati sul *bordo* del blocco (ossia, i lati che appartengono a *una sola* piastrella del blocco); la *lunghezza* del perimetro è data dal numero dei lati che lo compongono.

Esempio

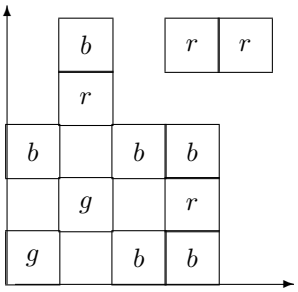
Nell'esempio sopra $\{A\}$, $\{B\}$, $\{C\}$, $\{A, B\}$, $\{B, C\}$, $\{A, B, C\}$, $\{A, C\}$ sono regioni, mentre $\{A, D\}$, $\{B, D\}$, $\{A, B, C, D\}$, $\{B, C, D\}$, etc., non lo sono. I blocchi sono $\{A, B, C\}$ e $\{D\}$. La lunghezza del perimetro del blocco a cui appartiene la piastrella A è 8.

Un *blocco omogeneo* è un blocco $\mathcal{A}$ in cui tutte le piastrelle accese hanno lo stesso colore α.

Il *blocco omogeneo di appartenenza* di Piastrella (x, y) è il blocco omogeneo che contiene Piastrella(x, y).

Esempio

Si consideri l'esempio seguente, dove i colori considerati sono r, g, b



Si noti che vi sono 2 blocchi: il blocco di appartenenza della piastrella posizionata in $(0, 0)$ contiene 10 piastrelle ed ha perimetro di lunghezza 30, mentre il blocco di appartenenza della piastrella in $(3, 4)$ conta 2 piastrelle e ha perimetro 6. Il blocco omogeneo di appartenza della piastrella posizionata in $(0, 0)$ contiene 2 piastrelle ed ha perimetro 8, mentre il blocco omogeneo di appartenenza della piastrella in $(2, 0)$ ha 2 piastrelle e perimetro di lunghezza 6.

Propagazione di colore

L'*intorno* della piastrella p è l'insieme di tutte le piastrelle circonvicine ad p e differenti da p.

È possibile definire *regole di propagazione* del tipo

$$k_1\alpha_1 + k_2\alpha_2 + \cdots + k_n\alpha_n \to \beta$$

dove n è un intero, $\alpha_1, \ldots, \alpha_n$ sono stringhe sull'alfabeto $\{a, b, \ldots, z\}$ tutte differenti tra loro, β è una stringa sull'alfabeto $\{a, b, \ldots, z\}$, k_i sono interi positivi la cui somma non supera 8.

La regola è *applicabile* a un a piastrella p se l'intorno di p contiene *almeno* k_i piastrelle di colore α_i per ogni $i \in \{1, \ldots, n\}$. L'effettiva *applicazione* della regola cambia in β il colore della piastrella p.

Si noti che l'applicabilità di una regola a una piastrella p non dipende dal colore di p, che potrebbe anche essere spenta.

Le regole di propagazione devono venire specificate esplicitamente attraverso un opportuno comando.

Le regole sono inserite dall'utente e vengono memorizzate in un elenco in ordine cronologico di inserimento (vale a dire, la regola inserita più recentemente è l'ultima dell'elenco).

L'elenco ordinato delle regole descrive come cambia il colore delle piastrelle. Qualora venga richiesto di propagare il colore su una piastrella, fra tutte le regole applicabili alla piastrella stessa viene selezionata ed applicata solo la prima di queste a comparire nell'elenco.

È possibile *propagare i colori su un blocco* in questo modo: per ogni piastrella colorata p appartenente al blocco si applica la prima regola dell'elenco applicabile a p rispetto all'intorno di p *cosí come si presenta all'inizio della procedura di propagazione sul blocco*. Dunque, eventuali cambi di colore di piastrelle del blocco appartenenti all'intorno non influiscono sulla colorazione di p.

Esempio

Riprendendo l'esempio precedente supponiamo che le regole inserite siano nell'ordine:

$$\begin{aligned} &2g + 1b \to z \\ &1g + 2b \to w \\ &1b + 1r \to y \\ &2b + 1r \to g \\ &1b + 1g + 1r \to t. \end{aligned}$$

Propaghiamo il colore su Piastrella$(1, 1)$: il suo intorno contiene 1 piastrella di colore g e 3 di colore b. Quindi la prima regola applicabile è la seconda dell'elenco. La sua applicazione colora Piastrella$(1, 1)$ col colore w. Supponiamo ora di voler colorare Piastrella$(3, 3)$, che attualmente è vuota. L'intorno contiene 2 piastrelle di colore b e 2 piastrelle di colore r, dunque la prima regola applicabile è la terza dell'elenco che conferisce a Piastrella$(3, 3)$ il colore y. Si noti che in questo caso anche la quarta regola è applicabile, ma non viene applicata perché è già applicabile la terza.

Riprendendo ora la propagazione iniziale dell'esempio propaghiamo il colore sul blocco di appartenenza di Piastrella$(1, 1)$. Le piastrelle che sono soggette a propagazione sono: Piastrella$(1, 1)$ che come già visto assumerà il colore w per applicazione della seconda regola, Piastrella$(2, 0)$ Piastrella$(2, 2)$, Piastrella$(3, 0)$ e Piastrella$(3, 2)$ che assumeranno il colore y per applicazione della terza regola. Le altre piastrelle del blocco restano invariate, poiché non si può applicare alcuna regola.

Durante l'esecuzione, ogni regola viene applicata un certo di numero di volte: definiamo *consumo* di una regola la quantità determinata dal numero totale di piastrelle a cui la regola è stata applicata dal momento della sua definizione. In dettaglio, siano $m \geq 0$ le operazioni di propagazione eseguite fino ad ora, e si denoti con $c_i(r)$ per ogni $i \in \{1, \ldots, m\}$ il numero totale di piastrelle a cui si è applicata la

regola r durante la i-esima operazione di propagazione. Allora il valore attuale del consumo della regola r è dato da $consumo(r) = \sum_{i=1}^{m} c_i(r)$.

Esempio

La propagazione sul blocco di appartenenza di Piastrella$(1, 1)$ mostrata nell'esempio precedente causa l'incremento di 4 unità dell'consumo della terza regola e di 1 della seconda.

4 Specifiche di progettazione

Si richiede di modellare la situazione con strutture dati opportune e di progettare algoritmi che permettano di eseguire efficientemente le operazioni elencate sotto. Le operazioni indicate con `[LUGLIO]`, o `[SETTEMBRE]` devono essere considerate solo per gli appelli di luglio o settembre, rispettivamente. Le altre operazioni devono essere considerate per ogni appello.

Le scelte di modellazione e di progettazione fatte devono essere discusse nella relazione, includendo l'analisi dei costi risultanti per le diverse operazioni.

Si richiede inoltre di predisporre una rassegna *esauriente* di esempi che potrebbero essere usati per testare il programma e che mettono in evidenza particolari caratteristiche del suo funzionamento (non solo casi tipici di input, ma anche casi limite e/o situazioni patologiche, oppure input che evidenzino la differenza di prestazioni tra le soluzioni progettuali scelte e altre meno interessanti).

Operazioni

- **colora**(x, y, α) Colora Piastrella(x, y) di colore α, qualunque sia lo stato di Piastrella(x, y) prima dell'operazione.
- **spegni**(x, y)

  Spegne Piastrella(x, y). Se Piastrella(x, y) è già spenta, non fa nulla.
- **regola**$(k_1, \alpha_1, k_2, \alpha_2, \ldots, k_n, \alpha_n, \beta)$

  Definisce la regola di propagazione $k_1\alpha_1 + k_2\alpha_2 + \cdots + k_n\alpha_n \to \beta$ e la inserisce in fondo all'elenco delle regole.
- **stato**(x, y)

  Stampa e restituisce il colore e l'intensità di Piastrella(x, y). Se Piastrella(x, y) è spenta, non stampa nulla.
- **stampa**

  Stampa l'elenco delle regole di propagazione, nell'ordine attuale.
- **blocco**(x, y)

  Calcola le stampa a somma delle intensità delle piastrelle contenute nel blocco di appartenenza di Piastrella(x, y). Se Piastrella(x, y) è spenta, restituisce 0.

- **bloccoOmog**(x, y)

  Calcola e stampa la somma delle intensità delle piastrelle contenute nel blocco omogeneo di appartenenza di Piastrella(x, y). Se Piastrella(x, y) è spenta, restituisce 0.

- **propaga**(x, y)

  Applica a Piastrella(x, y) la prima regola di propagazione applicabile dell'elenco, ricolorando la piastrella. Se nessuna regola è applicabile, non viene eseguita alcuna operazione.

- **propagaBlocco**(x, y)

  Propaga il colore sul blocco di appartenenza di Piastrella(x, y).

- **ordina**

  Ordina l'elenco delle regole di propagazione in base al consumo delle regole stesse: la regola con consumo maggiore diventa l'ultima dell'elenco. Se due regole hanno consumo uguale mantengono il loro ordine relativo.

- `[LUGLIO] [SETTEMBRE]`

  pista(x, y, s)

  Stampa la pista che parte da Piastrella(x, y) e segue la sequenza di direzioni s, se tale pista è definita. Altrimenti non stampa nulla.

- `[LUGLIO]`

  lung(x_1, y_1, x_2, y_2)

  Determina la lunghezza della pista più breve che parte da Piastrella(x_1, y_1) e arriva in Piastrella(x_2, y_2). Altrimenti non stampa nulla.

- `[SETTEMBRE]`

  intensità(x_1, y_1, x_2, y_2)

  Determina l'intensità minima tra le intensità di tutte le piste che partono da Piastrella(x_1, y_1) e arrivano in Piastrella(x_2, y_2). Se non vi è alcuna pista tra queste due piastrelle, non stampa nulla.

- `[SETTEMBRE]`

  perimetro(x, y)

  Calcola la lunghezza del perimetro del blocco di appartenenza della piastrella in (x, y).

Si noti che le operazioni richieste sono liberamente implementabili; in particolare, non vanno necessariamente intese come prototipi di funzioni.

5 Specifiche di implementazione

1. Il programma deve leggere dallo standard input una sequenza di righe (separate da a-capo), ciascuna delle quali corrisponde a una linea della prima colonna della Tabella 1, dove x, y, k_i sono interi e α, α_i, β, s sono stringhe finite di lunghezza *arbitraria*. I vari elementi sulla riga sono separati da uno spazio. Quando una riga è letta, viene eseguita l'operazione associata; le operazioni di stampa sono effettuate sullo standard output, e ogni operazione deve iniziare su una nuova riga.

2. Il programma deve contenere (almeno) cinque funzioni, corrispondenti alle prime cinque operazioni descritte nelle specifiche di progettazione e ai comandi indicati nella Tabella 1. Le funzioni devono avere queste segnature:

   `colora(p piano, x int, y int, alpha string)`

   `spegni(p piano, x int, y int)`

   `regola(p piano, r string)`

   `stato(p piano, x int, y int) (string, int)`

   `stampa(p piano)`

   Se queste funzioni non sono presenti nel programma o hanno una segnatura diversa da quella indicata, il progetto verrà estromesso dalla valutazione.

3. Il programma deve contenere:

   - la definizione di un tipo `piano` che rappresenta l'intero sistema, incluso l'insieme delle piastrelle e quello delle regole;
   - una funzione con segnatura:
     `esegui (p piano, s string)`
     che applica al sistema rappresentato da `p` l'operazione associata dalla stringa s, secondo quanto specificato nella Tabella 1.

Note

1. Non devono essere presenti vincoli sul numero di piastrelle.

2. Per semplicità si suppone che l'input sia sempre conforme alle specifiche di Tabella 1, per cui non è necessario controllare la correttezza dell'input.

3. Il formato di stampa di una regola r coincide con il formato di inserimento:

   $\beta\, k_1\, \alpha_1\, k_2\, \alpha_2 \cdots k_n\, \alpha_n$

   Se le regole fin ora inserite, in ordine di inserimento, sono $r_1, r_2, \ldots, r_m$, allora vanno stampate nel seguente formato:

   (
   r_1
   r_2
   $\vdots$
   r_m
)

	Linea di input	Operazione
	`C` x y α	**colora** (x, y, α)
	`S` x y	**spegni**(x, y)
	`r` β k_1 α_1 k_2 α_2 $\cdots$ k_n α_n	**regola** $(k_1, \alpha_1, k_2, \alpha_2, \ldots, k_n, \alpha_n, \beta)$
	`?` x y	**stato**(x, y)
	`s`	**stampa**
	`b` x y	**blocco**(x, y)
	`B` x y	**bloccoOmog**(x, y)
	`p` x y	**propaga**(x, y)
	`P` x y	**propagaBlocco**(x, y)
	`o`	**ordina**
	`q`	Termina l'esecuzione del programma
`[LUGLIO]`	`t` x y s	**pista**(x, y, s)
	`L` x y	**lung**(x_1, y_1, x_2, y_2)
`[SETTEMBRE]`	`t` x y s	**pista**(x, y, s)
	`i` x_1 y_1 x_2 y_2	**intensità**(x_1, y_1, x_2, y_2)
	`m` x y	**perimetro**(x, y)

Tabella 1: Specifiche del programma