Scuola Superiore di Udine
Esame di ammissione
Prova scritta di informatica
12/09/2006

Si risolvano i seguenti esercizi

1. Si vuole scrivere un programma per determinare l'area minima di un rettangolo che deve coprire un buco a forma di poligono rettangolare. Un poligono rettangolare è un poligono i cui angoli interni sono solo di 90 e 270 gradi. La forma del poligono rettangolare da considerare viene specificata con una sequenza di coppie (m, d) che descrivono una serie di segmenti concatenati, dove m è un intero e d una lettera (D,S,A,B). Assumendo che il poligono venga disegnato sul piano cartesiano, m è la lunghezza di un segmento e d è la direzione di quel segmento, dove D=destra, S=sinistra, A=alto e B=basso (Figura 1(a): (10,D) (10,A) (10,S) ...). Il programma, leggendo tale sequenza deve determinare l'area del minimo rettangolo (con lati paralleli agli assi cartesiani) che contiene il poligono rettangolare specificato. Ad esempio (2,D) (2,A) (1,S) (1,B) (1,S) (1,B) produce 4.

2. In un tunnel sotterraneo molto tortuoso che collega 2 palazzi vengono disposte delle luci in punti appositi da riuscire ad illuminare il tragitto fino alle luci immediatamente precedente e successiva. In corrispondenza delle luci e subito fuori dal tunnel ci sono dei commutatori. I commutatori esterni fanno commutare (acceso→spento, spento→acceso) le luci all'estremo corrispondente. Quelli interni invece fanno commutare le luci immediatamente precedente e successiva (se ci sono) rispetto alla luce corrispondente il commutatore stesso (Figura 1(b)). In questo modo le luci vengono lasciate normalmente spente e quando una persona vuole passare basta che continui a premere i commutatori che si ritrova lungo il cammino per avere illuminato sempre (e solo) il tratto che serve.

   Purtroppo ci sono persone malevole che con una torcia passano, muovendosi sempre in una stessa direzione, nel tunnel e non schiacciano tutti i commutatori che incontrano nel cammino per lasciare qualche luce accesa e molestare i successivi passanti. Per questo ogni tanto una guardia, munita di torcia, deve passare nel tunnel (muovendosi sempre in una stessa direzione) per ripristinare la configurazione tutta spenta.

   (a) Si sviluppi un algoritmo che determini la sequenza di azioni (schiacciare, non-schiacciare) che deve compiere la guardia sui vari commutatori incontrati, tenendo conto che anch'essa (a causa della configurazione del tunnel) non può vedere lo stato delle luci oltre la precedente e successiva. Si assuma di lavorare solo su configurazioni che possono essere state effettivamente prodotte da un malintenzionato (inclusa anche quella normale).

   (b) È possibile risalire alle azioni del malintenzionato, una volta sapute le azioni della guardia, senza sapere quali erano le direzioni di movimento dei 2?

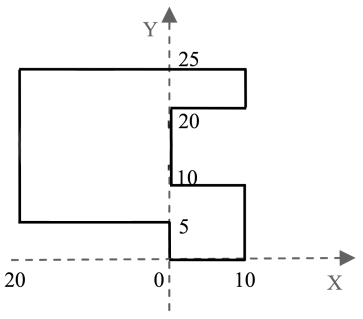
3. Si deve sviluppare un programma per scaricare informazioni storiche sui mercati finanziari da un servizio che fa pagare una determinata quota per le informazioni relative ad ogni giorno richiesto. Visto che si hanno già a disposizione i risultati di precedenti interrogazioni si vorrebbe risparmiare denaro evitando di richiedere informazioni duplicate con un programma che vada a modificare opportunamente le nuove richieste

prima di inoltrarle al servitore. Per semplificare la formalizzazione enumeriamo le date con numeri naturali. Codifichiamo una interrogazione con una sequenza di coppie $(data_{inizio}, data_{fine})$ con l'ovvio vincolo $data_{inizio} \leq data_{fine}$. Una sequenza di date $(a_0, b_0) \ldots (a_n, b_n)$ è detta *propria* se vale $a_{i+1} - b_i > 1$, per ogni $0 \leq i < n$. Si sviluppi un algoritmo che, date le codifiche **proprie** dello storico e della interrogazione desiderata, determina una codifica (preferibilmente quella propria) della richiesta (senza doppioni) da presentare al servitore. Ad esempio con storico (4,7) (10,13) e desiderato (1,5) (8,15) si produce (1,3) (8,9) (14,15).

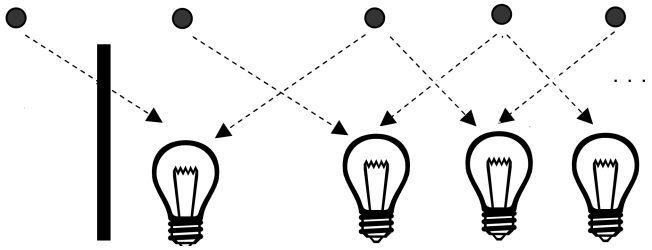
4. Molte tecniche sviluppate per la compressione di immagini si basano su una codifica chiamata "Quad Tree". Si codificano in questo modo immagini in bianco e nero quadrate in cui il numero di pixel del lato sia una potenza di 2. Se l'immagine è omogenea (stesso colore) si codifica con un bit a 1 seguito da un bit per il colore (per esempio un quadrato tutto nero sarà semplicemente 11, indipendentemente dalle sue dimensioni). Se l'immagine è eterogenea allora si utilizza un bit a 0 seguito dalle codifiche dei quadranti superiore-sinistro, superiore-destro, inferiore-sinistro, inferiore-destro, rispettivamente. Si sviluppi un algoritmo che partendo dalle codifiche di 2 immagini determina una codifica dell'immagine intersezione, cioè il risultato dell'AND logico delle 2 bitmaps (Figura 1(c)). Si assuma che le immagini in input abbiano la stessa dimensione (anche se la lunghezza della codifica non lo deve essere necessariamente). Si cerchi di generare codifiche minimali; per esempio 10 invece di 010101010.

Indicazioni su come presentare le soluzioni

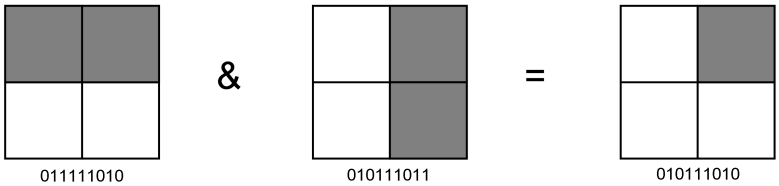
La descrizione dei programmi va fatta spiegando (succintamente) a parole le idee base e fornendo quindi una descrizione più formale. Questa descrizione può essere fatta nel formalismo che si ritiene più opportuno. È possibile utilizzare un linguaggio di programmazione standard (quale C, Pascal, Java, Scheme, Haskell, Prolog, ...) o più informalmente utilizzare un linguaggio di progetto (per esempio un linguaggio nella forma Pascal-like). Eventualmente si possono utilizzare anche i diagrammi di flusso. Nel presentare i programmi, si possono tralasciare aspetti non centrali, quali l'acquisizione dei dati, la stampa del risultato, il controllo della consistenza dei dati in ingresso. Il formato dei dati in "input" e in "output" può essere scelto a seconda della convenienza. Si raccomanda comunque di commentare i programmi proposti.



(a) un poligono rettangolare



(b) il tunnel



(c) esempio di intersezione

Figura 1: ausilio grafica