



UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dispense del Corso di Laboratorio di Fondamenti di Informatica II e Lab

Massimiliano Corsini, Federico Bolelli

Esercitazione 03: Backtracking

Ultimo aggiornamento: 23/03/2020

Backtracking: Ombrelloni

- Esercizio 1 (Ombrelloni):

Alcuni amici (k) trascorrono una giornata al mare. Giunti in spiaggia decidono di affittare un ombrellone ciascuno: tutti vogliono affittarne uno in prima fila, senza però essere vicini tra loro. La prima fila, contenente n ombrelloni, è tutta libera. Nel file `ombrelloni.c` si implementi la definizione della procedura ricorsiva `Ombrelloni`:

```
void Ombrelloni(int k, int n, unsigned i, bool *vcurr,  
                unsigned cnt, unsigned *nsol);
```

Backtracking: Ombrelloni

La procedura accetta i seguenti parametri:

- k : il numero di ragazzi da posizionare;
- n : il numero di posti disponibili in prima fila;
- i : la posizione attuale. Si noti che questo valore corrisponde al livello dell'albero di backtrack che la funzione corrente sta esplorando;
- $vcurr$: un array che indica lo stato degli ombrelloni in prima fila (ad esempio 1 = occupato, 0 = libero). All'inizio dovranno essere tutti liberi:

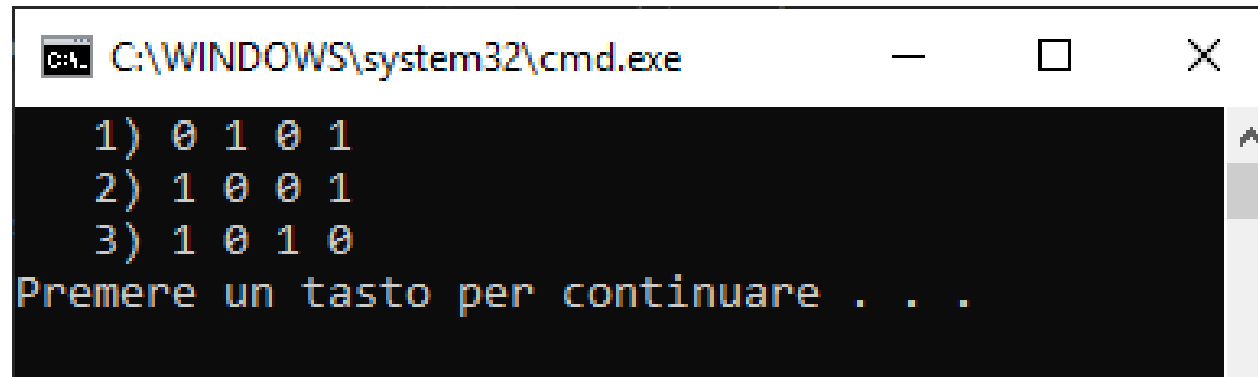
0	1	2	3	...	n-1	n
0	0	0	0	...	0	0

- cnt : un contatore che memorizza il numero di ragazzi posizionati nella soluzione corrente;
- $nsol$: il numero totale di soluzioni trovate ;

N.B è possibile implementare la funzione anche senza utilizzare il parametro cnt .

Backtracking: Ombrelloni

- La procedura ricorsiva deve visualizzare su standard output tutti i possibili modi in cui è possibile posizionare i k ragazzi nella fila di n ombrelloni. Ad esempio, con $k = 2$ e $n = 4$ l'output dovrà essere il seguente:



```
C:\WINDOWS\system32\cmd.exe

1) 0 1 0 1
2) 1 0 0 1
3) 1 0 1 0
Premere un tasto per continuare . . .
```

- Scrivete un opportuno `main()` di prova per testare la funzione.

Backtracking: Ombrelloni

- *Suggerimento:* cercate di visualizzare lo spazio di ricerca delle soluzioni (albero di decisione) prima di procedere con l'implementazione:
 - Quante e quali scelte posso fare ad ogni passo della ricorsione? Quanti figli ha/può avere ogni nodo dell'albero?
 - Quali sono le foglie dell'albero? Ovvero quali sono i casi di terminazione della ricorsione (casi base)?
- Essendo la prima esercitazione sul backtracking, per guidarvi nella soluzione dell'esercizio vi abbiamo specificato tutti i parametri della funzione `Ombrelloni`.
- Nelle future esercitazioni i prototipi delle funzioni conterranno solo i parametri relativi al problema e non quelli funzionali alla soluzione, che dovrete individuare autonomamente.

Backtracking: Babbo Natale

- Esercizio 2 (Babbo Natale):

Ogni anno che passa, Babbo Natale fatica sempre più a caricare la slitta dei regali che ha una portata massima di p kg. Per aiutarlo, nel file `babbonatale.c` si implementi la definizione della procedura ricorsiva `BabboNatale` che deve individuare tra n regali di peso `pacchi[i]` quali caricare per massimizzarne il numero totale, senza sforare la portata. La procedura deve avere il seguente prototipo:

```
void BabboNatale(int p, int const *pacchi, int n, unsigned i,  
                bool *vcurr, bool *vbest, unsigned *max, unsigned cnt,  
                int sum)
```

Backtracking: Babbo Natale

La procedura accetta i seguenti parametri:

- `p`: portata massima della slitta;
- `pacchi`: array dei pesi dei regali disponibili;
- `n`: dimensione dell'array `pacchi`;
- `i`: la posizione attuale. Si noti che questo valore corrisponde al livello dell'albero di backtrack che la funzione corrente sta esplorando;
- `vcurr`: un array che indica i regali attualmente caricati nella soluzione corrente (ad esempio 1 = caricato, 0 = NON caricato);
- ...

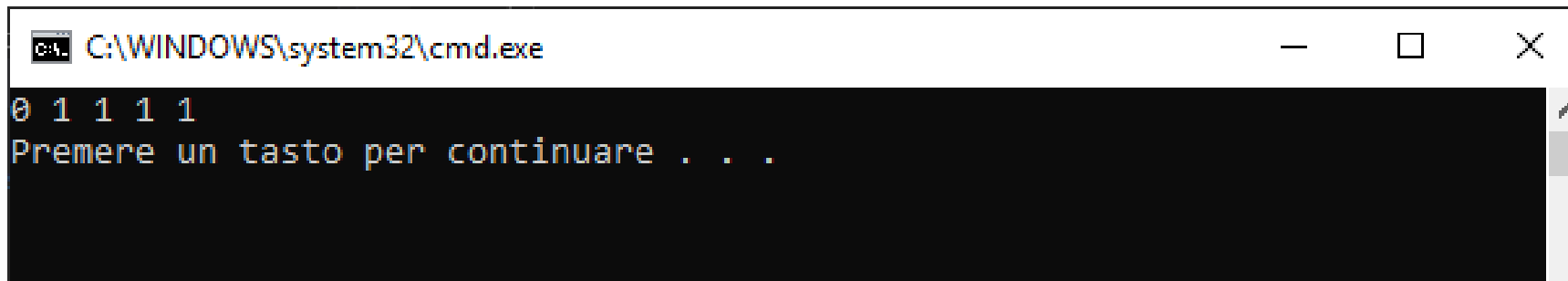
Backtracking: Babbo Natale

- ...
- `vbest`: un array che indica i regali caricati nella miglior soluzione fino ad ora trovata (ad esempio 1 = caricato, 0 = NON caricato);
- `max`: numero di regali caricati nella soluzione `best`;
- `cnt`: numero di regali caricati nella soluzione `vcurr`;
- `sum`: somma dei pesi dei regali caricati nella soluzione `vcurr`;

N.B. la funzione di backtracking potrebbe trovare la soluzione anche senza utilizzare i parametri `max`, `cnt` e `sum`.

Backtracking: Babbo Natale

- Mostrare su standard output la soluzione, ovvero la sequenza di regali che occorre caricare. Ad esempio, con $p = 20$ e $\text{pacchi} = \{ 10, 11, 1, 3, 3 \}$ l'output dovrà essere il seguente:



```
C:\WINDOWS\system32\cmd.exe
0 1 1 1 1
Premere un tasto per continuare . . .
```

- Dove 0 significa pacco NON caricato e 1 significa pacco caricato. La soluzione dell'esempio prevede quindi di caricare nell'ordine i pacchi di peso 11, 1, 3 e 3.
- Scrivete un opportuno `main()` di prova per testare la funzione.

Backtracking: Babbo Natale

- Per questo esercizio valgono le stesse note e gli stessi suggerimenti riportati per quello precedente.
- La soluzione trovata nell'esempio è l'unica possibile o esistono altre soluzioni equivalenti?

Modalità di Consegna

- Anche per questa esercitazione dovrete consegnare tutti gli esercizi inviando una mail a massimiliano.corsini@unimore.it e a federico.bolelli@unimore.it
- **Utilizzate solo l'account UNIMORE, ogni altra mail verrà ignorata.**
- **Specificate come oggetto [fdiii]. Non fdiii o [fdi II] o altre fantasiose soluzioni.**
- Ci raccomandiamo di inviare la mail ad entrambi gli indirizzi!
- Ogni esercizio dovrà essere svolto in file separati. I file dovranno avere i seguenti nomi:
`es1_ombrelloni.c`, `es2_babbo_natale.c`
- Ricordate che la consegna è facoltativa e non è necessario consegnare tutti gli esercizi.