Contents

1 Introduction

1.1 Succinct Data Structures

qua parlo di strutture succinte, minimal theoretical lower bound etc.

1.2 Classical select and rank queries

qua parlo di select e rank su sequenze di bit (quindi quelle classiche)

2 Notations

Qua introduco rank e select generalizzate su liste di interi:

$$rank_v(i) = \sum_{i=0}^{i-1} v_i$$

$$selectLast_v(i,u) = \begin{cases} \max\{j : rank_v(j) \leq i \wedge j < i\} & \{j : rank_v(j) \leq i \wedge j < i\} \neq \emptyset \\ -1 & \text{otherwise} \end{cases}$$

$$selectFirst_v(i,u) = \begin{cases} \min\{j : rank_v(j) \leq i \wedge j > i\} & \{j : rank_v(j) \leq i \wedge j > i\} \neq \emptyset \\ -1 & \text{otherwise} \end{cases}$$

3 Segment Tree

in questo capitolo mostro il funzionamento di un segment tree, come si costruisce e come si fa la select generalizzata (in modo non succinto, ovvero costruendo un segment tree che si riferisce all'intera lista di interi). volendo posso ampliarlo parlando delle altre operazioni che può fare ma che non servono nel nostro caso (ad esempio la modifica di un elemento).

3.1 Construction

3.2 selectLast and selectFirst

4 Two level organization of the information

qua mostro come costruire l'organizzazione a due livelli (simile a quella di jacobson) necessaria per fare rank e select succinte

4.1 First Level

$$S_i = \min_{ik^2 \le j < (i+1)k^2} \{\text{rank}_v(j)\} \qquad 0 \le i < \frac{n}{k^2}$$

$$T_i = \text{rank}_v(ik^2) \qquad 0 \le i < \frac{n}{k^2}$$

4.2 Second Level

$$B_i = \left(\min_{ik \le j < (i+1)k} \{\text{rank}_v(j)\} \right) - T_{\lfloor \frac{i}{k} \rfloor} \qquad 0 \le i < \frac{n}{k}$$

$$A_i = \text{rank}_v(ik) - T_{\lfloor \frac{i}{k} \rfloor} \qquad 0 \le i < \frac{n}{k}$$

5 Succinct rank, selectFirst and selectLast

qua mostro come fare rank e select generalizzate in modo succinto e con complessità $\mathcal{O}(\log n)$

6 An application of generalized rank/select queries: sequences of balanced parenthesis

qua introduco le sequence di parentesi ben bilanciate e descrivo in modo informale findclose, findopen e findenclose

6.1 Notations

Qua introduco formalmente le varie operazioni:

$$excess_v(i) = |\{j : \ j < i \wedge v_i = 1\}| - |\{j : \ j \le i \wedge v_i = -1\}|$$

$$findClose_v(i) = \begin{cases} \min\{j : \ j > i \wedge excess_v(j) = excess_v(i)\} & v_i = 1 \\ -1 & v_i = -1 \end{cases}$$

$$findOpen_v(i) = \begin{cases} \max\{j : \ j < i \wedge excess_v(j) = excess_v(i)\} & v_i = -1 \\ -1 & v_i = 1 \end{cases}$$

$$leftEnclose_v(i) = \begin{cases} \max\{j : \ j < i \wedge excess_v(j) + 1 = excess_v(i)\} & excess_v(i) > 0 \\ -1 & excess_v(i) = 0 \end{cases}$$

$$rightEnclose_v(i) = \begin{cases} \min\{j : \ j > i \wedge excess_v(j) + 1 = excess_v(i)\} & excess_v(i) > 0 \\ -1 & excess_v(i) = 0 \end{cases}$$

$$findEnclose_v(i) = (leftEnclose_v(i), rightEnclose_v(i))$$

6.2 Excess

qua spiego come calcolare l'eccesso di una posizione usando rank e select generalizzati:

$$excess_v(i) = \begin{cases} \sum_{j=0}^{i-1} v_j = rank_v(i) & v_i = 1 \\ \sum_{j=0}^{i} v_j = rank_v(i) - 1 & v_i = -1 \end{cases}$$

6.3 FindClose and findOpen

qua spiego come calcolare findclose e findopen con rank e select generalizzate:

$$findClose_v(a) = selectFirst_v(a, rank_v(a)) - 1 = selectFirst_v(a, excess_v(a)) - 1.$$

$$findClose_v(a) = selectLast_v(a, rank_v(a) - 1) = selectLast_v(a, excess_v(a)).$$

6.4 LeftEnclose and rightEnclose

Qua mostro come calcolare findenclose:

$$leftEnclose_v(i) = \begin{cases} selectLast_v(i, excess_v(i) - 1) & excess_v(i) > 0 \\ -1 & excess_v(i) = 0 \end{cases}$$

$$rightEnclose_v(i) = \begin{cases} selectFirst_v(i, excess_v(i) - 1) & excess_v(i) > 0 \\ -1 & excess_v(i) = 0 \end{cases}$$

7 Experiments

qua parlo principalmente dell'implementazione, di come è stata testata e del confronto con rank e select classiche.

7.1 succinctRankSelectDS

succinctRankSelectDS sarebbe l'implementazione della rank e select generalizzate (quindi capitolo 3,4,5)

7.1.1 Correctness test of the implementation

qua si parla del test di correttezza eseguito, non è molto utile questa parte ma è interessante il test di correttezza dell'altra struttura dati (capitolo 7.2.1)

7.1.2 Comparison with JacobsonDS

qua c'è il confronto con il rank di jacobson

Size	`SuccinctRankSelectDS`	`JacobsonDS`	`SlowJacobsonDS`
1 Kib	22.26	11.91	8.53
16 Kib	22.35	12.51	11.21
256 Kib	23.35	14.96	18.64
4 Mib	24.02	29.48	81.96
64 Mib	25.77	83.61	121.69
128 Mib	25.93	87.93	169.19

Table 1

7.2 SuccinctDS

succintDS è l'implementazione del rank e select generalizzato però applicato per la gestione di parentesi ben bilanciate (quindi qua non si parlerà di rank e select ma di findclose, findopen etc.)

7.2.1 Random generation of sequences of balanced parentheses

qua si parla della generazione casuale di sequenze di parentesi ben bilanciate, che è abbastanza interessante e per niente ovvio

7.2.2 Correctness test of the implementation

qua viene spiegato come è stato fatto un test di correttezza utilizzato il generatore di sequenze di parentesi ben bilanciate. volendo qua si potrebbe fare un altro capitolo dove faccio dei test di efficienza, ma che io sappia non ci sarebbero altre strutture dati (succinte) con cui fare un confronto.

8 Code

qua metto i codici, pensavo di mettere solo quelli che vengono esplicitamente menzionati nel capitolo experiments

9 Conclusions

conclusioni