Algoritmi e Strutture Dati

Esercitazione 4

Domenico Fabio Savo

RadixSort

Esercizio:

Mostrare il comportamento del RadixSort per ordinare alfabeticamente le seguenti parole:

```
A L B E R O
B O L L A
C A R
A B A C O
C A R O
```

Prestare particolare attenzione al fatto che le parole hanno lunghezze diverse, proponendo una soluzione generale per questo problema.

RadixSort

Una strategia per ordinare elementi con chiave di diversa lunghezza utilizzando il RadixSort è quella di **completare** le chiavi con dei simboli ad-hoc in modo da renderli tutti della medesima lunghezza.

Nel nostro caso possiamo fare le seguenti osservazioni:

Osservazione 1: alfabeticamente la parola ABACO precede la parola CAR dato che l'ordinamento alfabetico è effettuato a partire dalla lettera più a sinistra (la lettera A di ABACO precede alfabeticamente la lettera C di CAR) → le parole devono essere allineate a sinistra (ovvero il completamento deve essere effettuato aggiungendo simboli a destra delle parole)

RadixSort

Osservazione 2: alfabeticamente la parola CAR precede la parola CARO, perché?

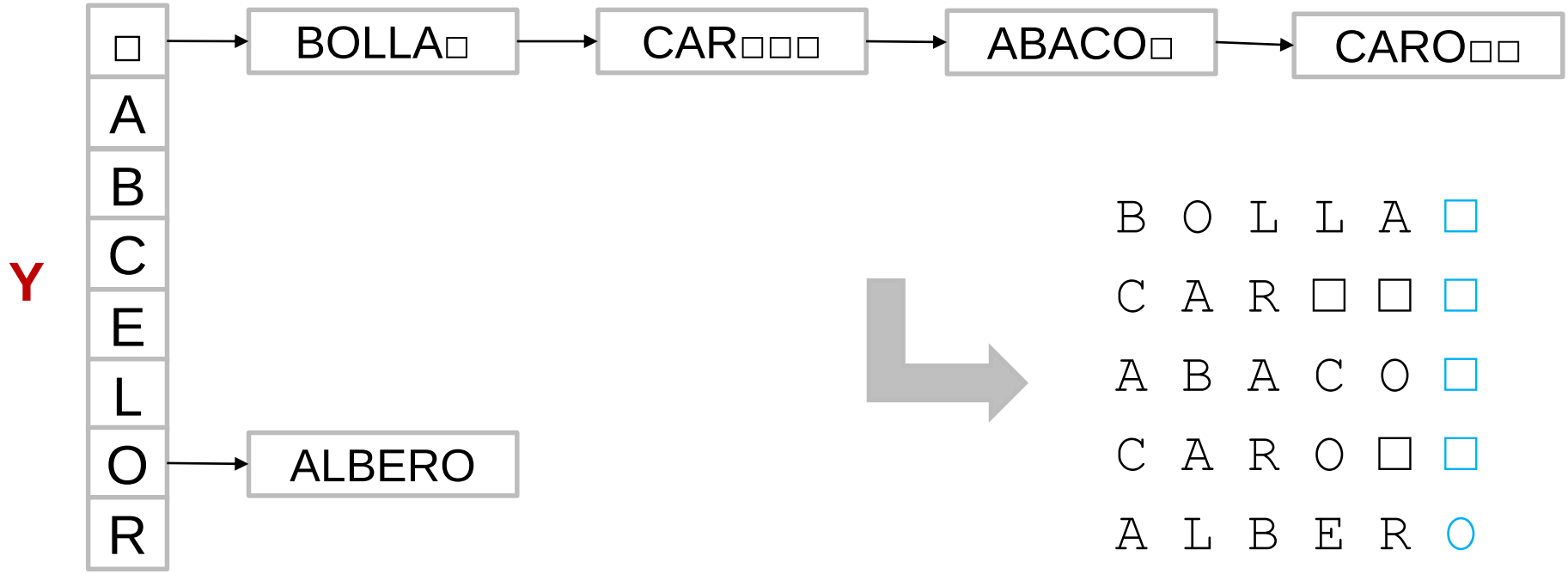
Si può immaginare che dopo la lettera R di CAR vi sia il simbolo blank space (che rappresentiamo qui con □) e che tale simbolo preceda alfabeticamente qualunque lettera dell'alfabeto.

La nostra lista di elementi diviene quindi:

A	L	B	E	R	O
B	O	L	L	A	□
C	A	R	□	□	□
A	B	A	C	O	□
C	A	R	O	□	□

RadixSort

A L B E R O
B O L L A □
C A R □ □ □
A B A C O □
C A R O □ □



B O L L A □
C A R □ □ □
A B A C O □
C A R O □ □
A L B E R O

RadixSort

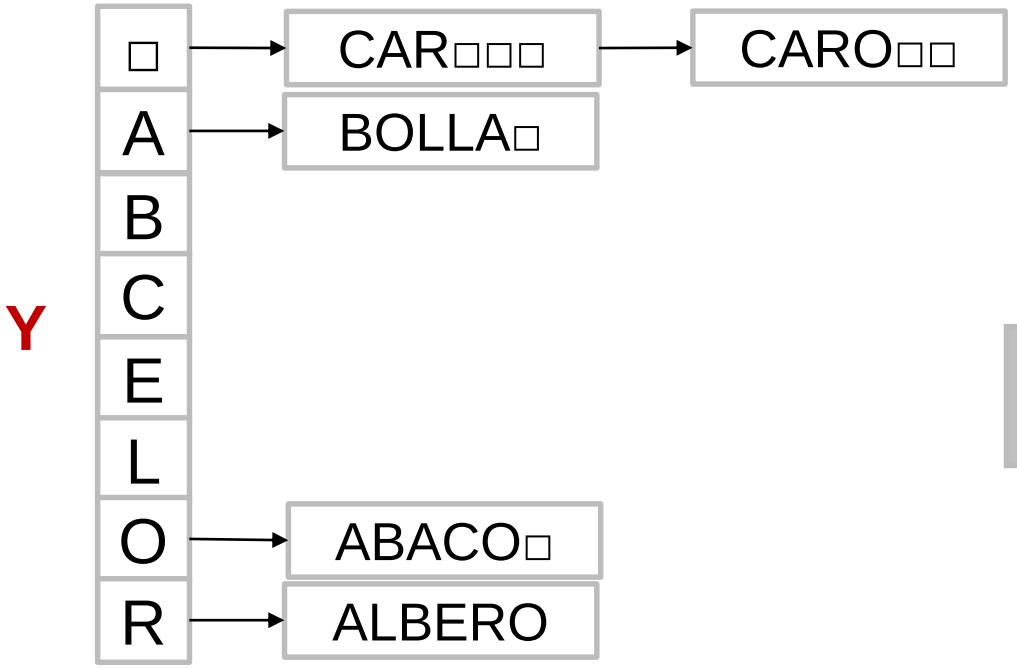
B O L L **A** □

C A R □ □ □

A B A C **O** □

C A R O □ □

A L B E **R** O



C A R □ □ □

C A R O □ □

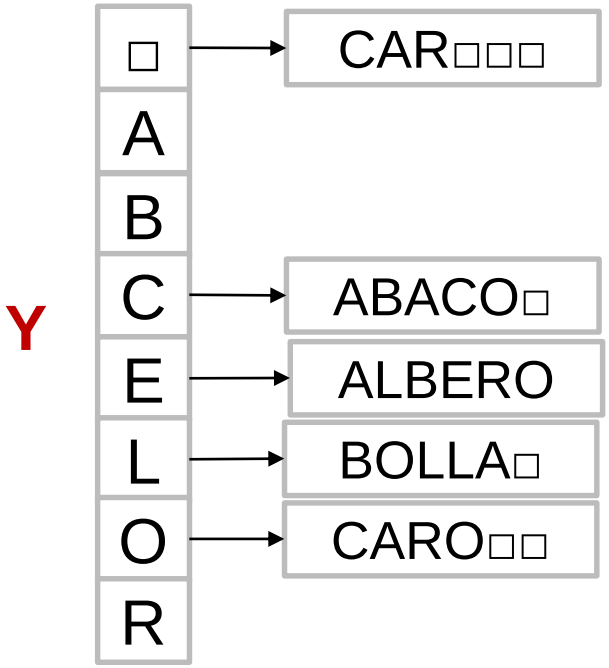
B O L L A □

A B A C O □

A L B E R O

RadixSort

C A R □ □ □
C A R O □ □
B O L L A □
A B A C O □
A L B E R O



C A R □ □ □
A B A C O □
A L B E R O
B O L L A □
C A R O □ □

RadixSort

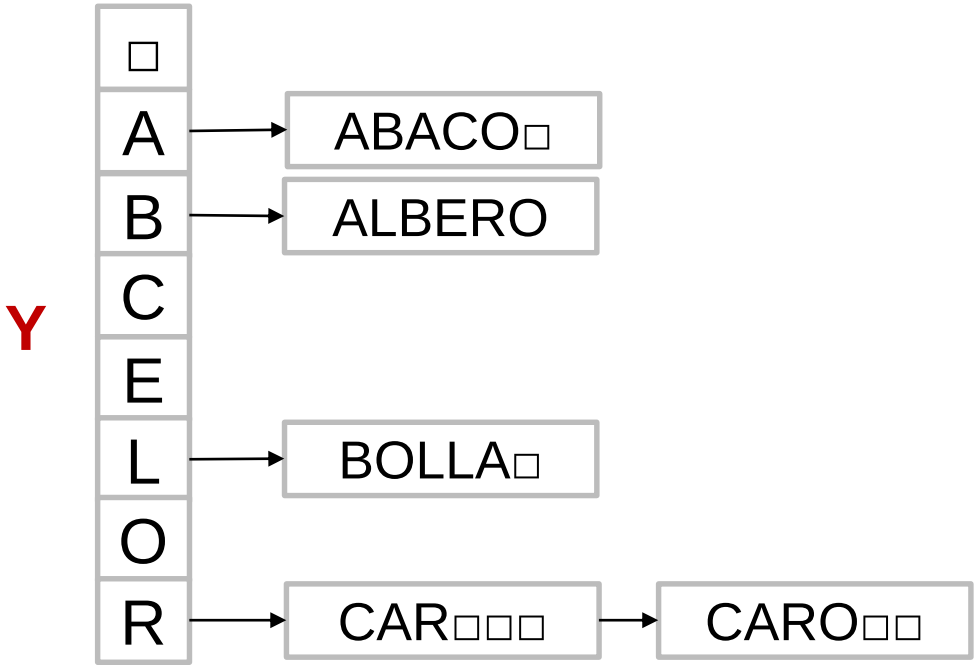
C A **R** □ □ □

A B **A** C O □

A L **B** E R O

B O **L** L A □

C A **R** O □ □



A B A C O □

A L B E R O

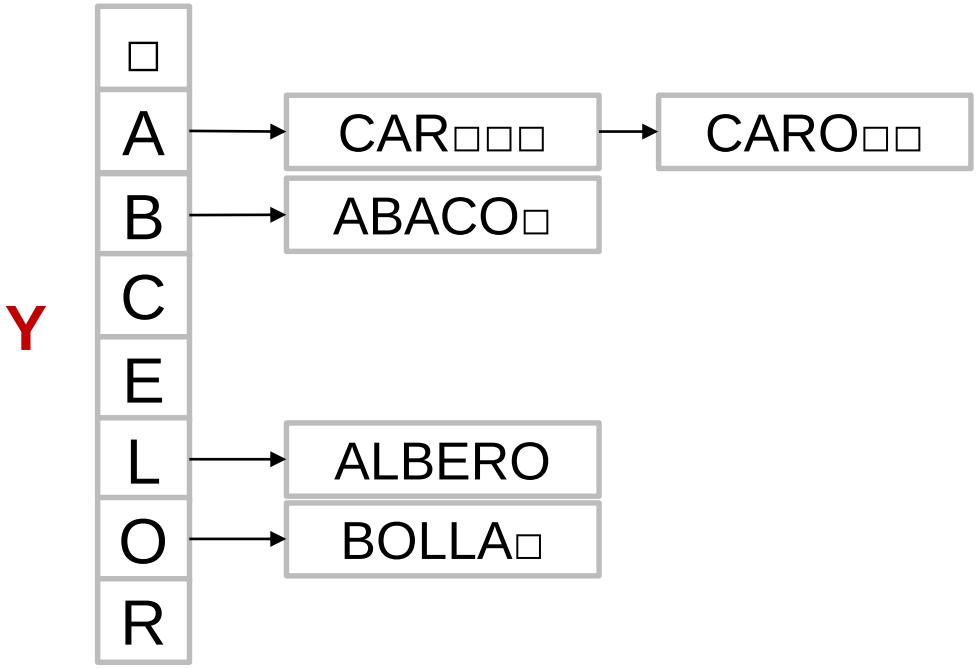
B O L L A □

C A R □ □ □

C A R O □ □

RadixSort

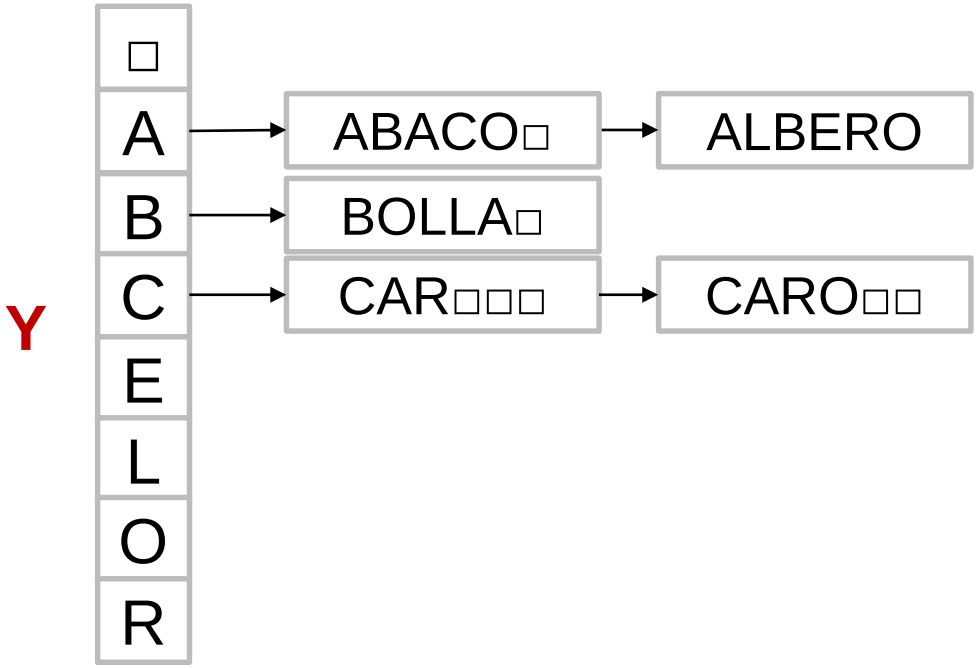
A B A C O □
A L B E R O
B O L L A □
C A R □ □ □
C A R O □ □



C A R □ □ □
C A R O □ □
A B A C O □
A L B E R O
B O L L A □

RadixSort

C A R □ □ □
C A R O □ □
A B A C O □
A L B E R O
B O L L A □



FINE!

A B A C O □
A L B E R O
B O L L A □
C A R □ □ □
C A R O □ □

fogliePari

Esercizio:

Scrivere l'algoritmo

foglieePari(nodo r) → *Boolean*

che preso in ingresso un albero binario radicato in r, verifica se tutte le foglie dell'albero contengono valori pari.

fogliePari

Soluzione:

```
Algoritmo fogliePari(nodo r)→ Boolean
if(r = null) then return true;
if(r.figlio_sx() = null && r.figlio_dx() = null) then
      {//r non ha figli quindi è una foglia
       if(r.elem %2 = 0) then return true;
       else return false;
      }
else return  fogliePari(r.figlio_sx) &&
             fogliePari(r.figlio_dx);
```