Laboratorio di Algoritmi

Progetto "Insiemistica" (settembre 2021)

Nota: La scadenza del progetto è fissata per lunedì 13 settembre **compreso**.

Nota: Si consiglia di consultare sulla pagina web il documento che riporta le avvertenze utili per lo svolgimento del progetto. Si consiglia anche di verificare di tanto in tanto gli aggiornamenti a questo documento, che riporteranno la correzione di eventuali errori e le risposte ai dubbi degli studenti.

Il problema Si vuole realizzare un "calcolatore insiemistico", che gestisca insiemi di parole, definiti elencando esplicitamente le parole che li compongono, ed esegua poi su tali insiemi le classiche operazioni elementari di unione, intersezione, differenza e differenza simmetrica, oppure operazioni composte, che combinano gerarchicamente quelle elementari.

Il calcolatore deve anche rispondere a interrogazioni riguardo lo stato degli insiemi stessi (se sono già stati definiti oppure no) e la relazione fra parole e insiemi (se una parola data appartenga o no a un insieme dato). Infine, deve fornire informazioni riassuntive sullo stato degli insiemi al termine dell'elaborazione.

Terminata l'elaborazione e l'analisi degli insiemi, il calcolatore deve procedere con un'analisi delle parole e delle loro relazioni in base a un'opportuna definizione di "adiacenza" introdotta nel seguito.

Il progetto Il programma da realizzare carica i dati da un file di testo il cui nome va fornito nella linea di comando. Ogni riga del file contiene un'istruzione che il calcolatore deve eseguire. Sono possibili i seguenti tipi di istruzione:

1. assegnamenti di insiemi, a loro volta distinti in:
   - assegnamenti espliciti;
   - assegnamenti con operazioni insiemistiche;
2. interrogazioni sullo stato di un insieme;
3. interrogazioni sul contenuto di un insieme;
4. interrogazioni sulla relazione fra una parola e un insieme.

L'assegnamento esplicito di un insieme rispetta il seguente formato: comincia con il nome dell'insieme (una lettera maiuscola dell'alfabeto inglese, da `A` a `Z`), prosegue con l'operatore simbolico di assegnamento `:=` e si conclude con l'elenco dettagliato delle parole da includere nell'insieme, racchiuso fra parentesi graffe. Le parole elencate in un assegnamento esplicito sono tutte diverse tra loro. Nomi di insiemi, operatori, parole e parentesi sono tutte separate tra loro da spazi bianchi. L'assegnamento occupa una sola riga. Se l'insieme compare per la prima volta in un assegnamento, si dice che viene *definito*; se invece era già comparso in precedenza, le parole indicate nell'assegnamento sostituiscono il contenuto originale dell'insieme. Per esempio, se il file delle istruzioni comincia con le tre righe:

```
C := { prova temporanea }
A := { naso prova casa }
C := { casa velo vaso pilota cane }
```

l'insieme `C` viene definito come composto dalle 2 parole `prova` e `temporanea`. Quindi, l'insieme `A` viene definito come composto dalle 3 parole `naso`, `prova` e `casa`. Infine, l'insieme `C` che era già definito, viene modificato assegnandogli le 5 parole `casa`, `velo`, `vaso`, `pilota` e `cane`. Il numero di parole in un assegnamento, e più in generale in un insieme, può essere grande a piacere. Ogni parola è formata da un massimo di 10 caratteri, che sono tutti lettere minuscole dell'alfabeto inglese, da `a` a `z`[1].

A seguito di un'istruzione di assegnamento esplicito, il programma deve stampare a video una riga di testo che riporta il nome dell'insieme seguito dalle parole chiave `defined with` oppure `assigned with`, dal numero di elementi e dalla parola chiave `elements`, secondo che l'insieme stesso viene definito o semplicemente modificato dall'assegnamento. In corrispondenza alle tre istruzioni precedenti, quindi, il programma deve stampare:

```
C defined with 2 elements
A defined with 3 elements
C assigned with 5 elements
```

Gli assegnamenti che comportano l'esecuzione di un'operazione insiemistica cominciano, come quelli espliciti, con il nome dell'insieme e con l'operatore `:=`, ma proseguono con un'espressione insiemistica. Per convenzione, un'espressione insiemistica è sempre racchiusa fra parentesi tonde, così da distinguerla chiaramente da una sequenza di parole. L'espressione può essere:

- elementare, cioè costituita dal semplice nome di un insieme (ad es., `( A )`);
- composta, cioè costituita da un'operazione binaria, che opera su due insiemi per generarne un terzo, riportata in notazione infissa, cioè con l'operatore che descrive l'operazione posto in mezzo ai due operandi, che sono a loro volta espressioni insiemistiche (elementari o composte), racchiuse fra le relative parentesi tonde (ad es., `( ( A ) diff ( C ) )`).

Nomi di insiemi, parentesi e operatori sono tutti separati da spazi bianchi.

Le operazioni insiemistiche elementari e i relativi operatori sono:

1. l'unione, che è descritta dall'operatore `union` e genera l'insieme degli elementi che appartenevano ad almeno uno di due insiemi dati;
2. l'intersezione, che è descritta dall'operatore `inters` e genera l'insieme degli elementi che appartenevano ad entrambi gli insiemi dati;
3. la differenza, che è descritta dall'operatore `diff` e genera l'insieme degli elementi che appartevano al primo insieme dato, ma non al secondo;
4. la differenza simmetrica, che è descritta dall'operatore `symdiff` e genera l'insieme degli elementi che appartenevano ad uno solo dei due insiemi dati.

Eseguite tutte le operazioni e assegnato il risultato all'insieme dato, il programma stampa lo stesso messaggio richiesto per gli assegnamenti espliciti. Per esempio, le istruzioni:

```
F := ( ( A ) inters ( C ) )
C := ( ( ( A ) diff ( C ) ) union ( F ) )
```

[1] Siccome il limite sulla lunghezza delle parole è stato introdotto solo per semplificare le operazioni di lettura, le analisi di complessità devono considerare infinito l'insieme delle parole possibili, anche se il limite garantisce che ce ne siano al massimo 26^{10}.

a seguito delle istruzioni viste in precedenza devono provocare la stampa dei seguenti messaggi:

```
F defined with 1 elements[2]
C assigned with 3 elements
```

perché l'insieme `F` compare per la prima volta e contiene solo la parola `casa`, mentre l'insieme `C` viene modificato in modo da contenere le tre parole `naso`, `prova` e `casa`.

L'interrogazione sullo stato di un insieme osserva il formato seguente: alla parola chiave `isdef` segue il nome dell'insieme su cui si vuole eseguire l'interrogazione. Il programma deve stampare il nome dell'insieme, seguito da `is defined` oppure `is NOT defined` secondo che l'insieme indicato sia stato definito oppure no in una precedente istruzione. Per esempio, le istruzioni:

```
isdef M
isdef C
```

devono provocare la stampa dei seguenti messaggi:

```
M is NOT defined
C is defined
```

L'interrogazione sul contenuto di un insieme osserva il formato seguente: alla parola chiave `print` segue il nome dell'insieme investigato. Il programma deve stampare il nome dell'insieme, seguito dal simbolo `=` e dall'elenco (racchiuso fra parentesi graffe) delle parole contenute nell'insieme, in ordine lessicografico. Se l'insieme non è definito, bisogna stampare a video il nome dell'insieme e l'indicazione `is NOT defined`. Per esempio, le istruzioni:

```
print C
print X
```

a valle delle precedenti inducono la stampa dei seguenti messaggi:

```
C = { casa naso prova }
X is NOT defined
```

Infine, l'interrogazione sulla relazione fra una parola e un insieme comincia con la parola chiave `isin`, seguita dal nome dell'insieme e dalla parola presi in considerazione. Il programma deve stampare la parola, seguita da `is in` e dal nome dell'insieme oppure da `is NOT in` e dal nome dell'insieme, secondo che l'insieme contenga o no la parola data. Se l'insieme non è definito, per convenzione la parola non gli appartiene. Per esempio, le istruzioni:

```
isin vaso C
isin naso C
```

eseguite dopo le precedenti, richiedono la stampa dei seguenti messaggi:

```
vaso is NOT in C
naso is in C
```

Conclusa l'elaborazione di tutte le istruzioni, il programma deve riassumere lo stato finale del sistema, stampando gli insiemi finali definiti. Gli insiemi vanno stampati in modo lessicografico, vale a dire che un insieme ne precede un altro se la sua prima parola (in ordine alfabetico) precede la prima dell'altro, mentre lo segue in caso contrario; se le rispettive prime parole sono identiche, si procede confrontando le seconde, e così via. Se uno dei due insiemi termina e l'altro no, il primo precede

[2]Per semplicità (anche mia nella correzione), lasciamo perdere la correttezza grammaticale.

il secondo. Nel caso di insiemi identici, vale l'ordine alfabetico dei loro nomi. Per ogni insieme, si stamperà una riga che comincia con il nome dell'insieme seguito da `=` e dall'elenco delle sue parole in ordine alfabetico, racchiuso fra parentesi graffe. Sempre riferendosi all'esempio usato in precedenza:

```
F = { casa }
A = { casa naso prova }
C = { casa naso prova }
```

Quindi, il programma stampa una riga con il numero di parole complessivamente presenti negli insiemi finali seguito dalla parola chiave `parole` e una riga con il numero di occorrenze di parole presenti negli insiemi finali seguito dalla parola chiave `occorrenze`. Nel caso delle occorrenze, la stessa parola in insiemi diversi conta una volta per ciascun insieme.

```
3 parole
7 occorrenze
```

Passando dall'analisi degli insiemi a quella delle parole, il programma stamperà l'elenco in ordine alfabetico di tutte le parole introdotte durante l'elaborazione, cioè non solo quelle incluse negli insiemi finali, ma anche quelle eventualmente cancellate durante il processo. Ogni parola va stampata su una riga diversa. Nel nostro esempio:

```
cane
casa
naso
pilota
prova
temporanea
vaso
velo
```

Infine, il programma definirà per ogni parola un vettore che riporta il numero di occorrenze di ciascuna lettera nella parola stessa. In base a questi vettori, definirà una "distanza" tra parole sulla base della distanza L_1, ovvero "di Manhattan", fra i due vettori (cioè la somma dei valori assoluti delle differenze elemento per elemento). Considerando *adiacenti* due parole tali che la distanza reciproca sia non superiore a $\delta = 3$, il programma costruirà il relativo grafo che ammette le parole come vertici e le relazioni di adiacenza come lati e ne determinerà le componenti connesse. Quindi stamperà a video il numero di tali componenti seguito dalla parola chiave `componenti` e i loro elementi, una componente per riga, in ordine di cardinalità decrescente e (in caso di parità) in ordine lessicografico. Questo ordine va inteso come descritto sopra per gli insiemi di parole usati dal calcolatore insiemistico; infatti, si tratta sempre di insiemi di parole, anche se non sono gli stessi e potrebbero essere rappresentati in modo diverso. Gli elementi di ciascuna componente andranno stampati in ordine alfabetico. Nel solito esempio, le coppie di parole adiacenti sono (`naso`,`vaso`) e (`prova`,`vaso`). Di conseguenza, le componenti connesse sono sei: la più grande contiene (nell'ordine) le 3 parole `naso`, `prova` e `vaso`, le altre contengono una sola parola ciascuna, rispettivamente `cane`, `casa`, `pilota`, `temporanea` e `velo`.

```
6 componenti
naso prova vaso
cane
casa
```

```
pilota
temporanea
velo
```

Chiarimenti

Ha creato un po' di scompiglio la nota sul fatto che le parole sono di lunghezza ≤ 10, ma il loro numero potenziale va considerato infinito, anche se a rigore non può superare 26^{10}, dato che l'alfabeto inglese ha 26 lettere. Lo scopo della duplice osservazione è evitare due possibilità incresciose dal punto di vista della valutazione:

1. che qualcuno consideri il numero di parole potenziali limitato, cioè $O(1)$ e ne deduca (correttamente) che gli insiemi di parole hanno cardinalità in $O(1)$, e quindi tutte le operazioni richiedono tempo e spazio in $O(1)$, in questo modo ammazzando l'analisi di complessità;

2. che qualcuno consideri la lunghezza delle parole non maggiorabile a priori e quindi si senta in dovere di complicare la procedura di lettura per gestire lunghezze qualsiasi, in questo modo introducendo complicazioni che hanno poco a che fare con i temi del corso.