Opracowanie: exc/boi-003
Excursion

HISTORIA:

dokument systemu SINOL 1.3.1

1 Analiza

1.1 Podobne zadanie

Niemal równoważne zadanie wystąpiło na II etapie polskiej VIII Olimpiady Informatycznej w roku 2001 (Spokojna Komisja).

To jest zadanie na spełnialność formuł 2-CNF. Jeśli oznaczymy przez a_i zmienną, oznaczającą „wycieczka jedzie przez miasto o numerze i”, to naszym celem jest znalezienie wartościowania tych zmiennych spełniających formułę

$$(l_1^1 \vee l_1^2) \wedge (l_2^1 \vee l_2^2) \wedge \ldots \wedge (l_n^1 \vee l_n^2)$$

gdzie l_j^1, l_j^2 to życzenia j-tego turysty wyrażone jako literały, tj. postaci a_i lub $\neg a_i$.

Przypadki turystów:

- $a \vee b$, $a \neq b$ – przekształcamy równoważnie: ($\spadesuit$) $\neg a \Rightarrow b$, ($\spadesuit$) $\neg b \Rightarrow b$, ($\clubsuit$) $\neg(\neg a \wedge \neg b)$.
- $\neg a \vee b$, $a \vee \neg b$, $\neg a \vee \neg b$ – są analogiczne do poprzedniego
- ($\dagger$) $a \vee a$ – to oznacza że na pewno musimy wziąść a
- $a \vee \neg a$ – to ograniczenie jest zawsze prawdziwe i możemy je pominąć.

Z powyższych możemy korzystać przy rozwiązywaniu zadania.

Rozważmy graf w którym wierzchołki odpowiadają literałom. Dla każdego miasta mamy dwa wierzchołki. Ponumerujmy wierzchołki $1, \ldots, 2m$: a_i jako $2i - 1$, a $\neg a_i$ jako $2i$. Mamy wybrać po jednym wierzchołku z każdej pary.

Niektóre wierzchołki, jak wynika z ($\dagger$), mogą być „pewniakami”.

Możemy dodać krawędzie na podstawie ($\clubsuit$), otrzymamy wtedy graf konfliktów wierzchołków, tj. połączonych wierzchołków nie możemy wybrać.

Możemy zrobić też graf skierowany „wnioskowania”, na podstawie ($\spadesuit$). Otrzymany graf interpretujemy tak: jeśli wybierzemy dany wierzchołek, to musimy też wszystkich jego następników.

2 Rozwiązanie wzorcowe

Pierwszym nasuwającym się rozwiązaniem jest backtracking. Rozpatrujemy miasto i, dla którego jeszcze nie powzięliśmy decyzji. Wybieramy pierwszą z opcji (a_i) i na podstawie grafu wnioskowania podejmujemy wymuszone decyzje; sprawdzamy przy tym, czy taki wybór nie pociągnie za sobą dwóch sprzecznych decyzji. Jeśli tak, to rozpatrujemy drugą z opcji ($\neg a_i$). I tak z nawrotami.

Uważnie się przyglądając można zauważyć, że nawroty są niepotrzebne. Załóżmy że w trakcie działania algorytmu rozpatrujemy kolejne miasto, dla którego nie podjęliśmy decyzji. Zauważmy najpierw, że żaden z innych wierzchołków, których wybranie pociąga za sobą decyzję w sprawie tego miasta, nie jest w konflikcie z żadnym z wcześniej podjętych wyborów. Istotnie, nie może tak być, gdyż już wcześniej rozpatrywaliśmy wszystkie miasta, które są w konfilkcie z danymi wyborami. Stąd wniosek, że na decyzję, którego dokonujmy, nie mają wpływu żadne poprzednie wybory; ani też ta decyzja nie wpłynie na późniejsze decyzje. (Decyzja, to to, co podejmujemy; wybory, to ponadto to, do czego nas owe decyzje zmuszają).

A zatem wystarczy taki jednopoziomowy nawrót, w razie decyzji prowadzącej do sprzeczności. Można ten algorytm nazwac „zachłannym".

Takie rozwiązanie bez większych trudów można zaimplementować w pesymistycznym czasie $O((m+n)*m)$, gdzie m to liczba miast (zmiennych), a n – liczba turystów (ograniczeń).

Można też zrobić to sprytniej:

1. generujemy graf wnioskowania
2. w takim grafie szukamy silnych spójnych składowych,
3. jeśli do jeden silnej spójnej składowej należą dwie sprzeczne decyzje dotyczące tego samego miasta, to oznacza że możemy pominąć w rozważaniach wszystkie wierzchołki z tej s.s.składowej i te dla których istnieje ścieżka do rozważanej s.s.składowej (wybór któregokolwiek z tych wierzchołków automatycznie prowadzi do sprzeczności)
4. sortujemy topologicznie pozostałe silne spójne składowe, i rozważamy je w tej kolejności,
   (a) jeśli aktualna s.s.składowa nie została odrzucona to wybieramy wszystkie należące do niej wierzchołki
   (b) dla każdego wybranego w poprzednim kroku wierzchołka, odrzucamy s.s.składową, która zawiera wierzchołek opozycyjny doń, (oraz składowe z których istnieje do takiego ścieżka)
5. jeśli dokonano n wyborów odpowiadamy TAK, wpp NIE

Jest to rozwiązanie o złożoności czasowej $O(n+m)$.

Implementacja znajduje się w `prog/exc.pas`

Uwaga: z uwagi na spore zapotrzebowanie na pamięć wzorcowa implementacja wymaga kompilatora FPC, a nie BP7.

3 Testy

Testy zostały przekonwertowane z testów do zadania Spokojna Komisja, plus wprowadzono do nich niekiedy podróżników o życzeniach dotyczących tego samego miasta (co nie występowało w tamtym zadaniu).

- `exc0.IN` (ϵ sek.) Test przykładowy
- `exc0a, 0b, 0c, 0d.IN` (ϵ sek.) Proste testy testujące szczególne przypadki – nie do punktacji
- `exc1.IN` (ϵ sek.) Mały test poprawnościowy
- `exc2.IN` (ϵ sek.) Mały test poprawnościowy
- `exc3.IN` (ϵ sek.) Średni test poprawnościowy
- `exc4a.IN` (ϵ sek.) Średni test z odpowiedzią NIE, dla którego backtraking działa wykładniczo.

- `exc4b.IN` (ϵ sek.) Średni test, wydajnościowy, dla którego komisja istnieje.
- `exc5a.IN` (ϵ sek.) Średni test z odpowiedzią NIE, dla którego backtraking działa wykładniczo.
- `exc5b.IN` (ϵ sek.) Średni test, dla którego komisja istnieje.
- `exc6a.IN` ($0.25s$ sek.) Duży test z odpowiedzią NIE, dla którego backtraking działa wykładniczo.
- `exc6b.IN` ($0.25s$ sek.) Duży test, dla którego komisja istnieje.
- `exc7.IN` ($0.25s$ sek.) Maksymalnej wielkości test.
- `exc8, 8a, 8b.IN` ($0.5s$ sek.) Grupa testów, dla odróżnienia rozwiązania $O((m+n)*m)$ od liniowego

Testy powinny być grupowane.