알고리즘 HW #4

컴퓨터공학부
2013-11431 정현진

1. BoyerMooreHorspool 알고리즘으로 스트링 매칭을 하려한다. 총 10,000개의 문자로 이루어진 텍스트에서 “camouflage"라는 패턴을 검색하려 할 때 다음의 질문에 답하라. (asymptotic number가 아니고, 정확한 숫자를 요구함.)

1.1 가장 운이 좋으면 텍스트에서 몇 개의 문자를 살펴보고 끝날 수 있는가?
1.2 가장 운이 나쁘면 텍스트에서 몇 개의 문자를 살펴보고 끝날 수 있는가?

주어진 문자열을 A[]라고 하고, 비교하려는 문자열을 P[]라고 하자. 이 문제의 경우 A는 A[0]에서부터 A[9999]까지 존재하고, P는 P[0] = ‘c’ 에서부터 P[9] = ‘e’ 로 존재한다.

1.1
가장 운이 좋은 경우는 A가 A[0]~A[9]까지 P와 같은 “camouflage“ 라는 문자열로 이루어져 있을 경우이다. 이 경우 우선 A[9]와 P[9]를 비교하는데 둘이 모두 ‘e’로 같으므로, 앞의 문자열을 비교한다. 모두 같으므로 A[8] <-> P[8] 에서 A[0] <-> P[0]까지 비교하게 되므로 총 9번을 비교하고, 이전의 1번을 더하면 10번이 된다. 그리고 P와 같은 문자열이 매칭되었으므로 종료한다. 즉 운이 가장 좋은 경우 10개의 문자를 살펴보고 끝날 수 있다.

1.2
가장 운이 나쁜 경우는 A가 첫 글자는 P에 없는 문자, 다른 글자는 모두 P와 같은 문자열로 반복되는 경우이다. 문자열은 하나 예시로 들면 “kamouflage”인데, 이러한 문자열로 A[0]~A[9], A[10]~A[19], … , A[9990]~A[9999]까지 모두 채워져 있는 경우이다. 이 경우의 비교를 생각해보면, 우선 A[9]와 P[9]를 비교한 뒤 같으므로 앞으로 가면서 비교하는데, 첫 글자까지 비교한 후 두 문자열이 다르다는 것을 알게 된다. 그 뒤 P아 10칸 뒤로 가서 같은 작업을 반복하다가 P와 A[9999]~A[9990]을 비교한 뒤 맞는 문자열을 찾지 못하고 끝나게 된다. 이러한 경우 A[0]부터 A[9999]까지의 모든 문자를 다 살펴보아야 하므로 최악의 경우 살펴보아야 할 문자의 개수는 총 10000개이다.

2. 2SAT은 다음과 같이 정의된다. 2SAT∈ P 임을 증명하라.

> **2SAT = { Φ | Φ is a satisfiable formula in CNF with at most 2 literals per clause }**
>
> **Hint: A clause x ∨ y can be written as ¬x ⇒y or as ¬y ⇒x . Consider a graph whose vertices are variables and their complements, and where there is a directed edge from ¬x to y if x and y are in the same clause.**

힌트대로 $x \vee y$인 clause를 $\overline{x}$->y의 edge와 $\overline{y}$->x의 edge를 추가하는 식으로 그래프를 구성해보자. clause의 literal이 1개만 존재하는 경우는 x ∨ x = $\overline{x}$ -> x 같은 형식으로 나타낼 수 있다.
이 때 그래프의 어떠한 edge에서도 ‘참’->‘거짓’이 되는 경우가 없어야 satisfiable함을 알 수 있다. 다르게 말하면, edge의 시작점이 참인 경우 도착점도 항상 참이어야 한다.

unsatisfiable한 조건은 다음과 같다.

> 임의의literal ‘x’가 있을 때, x -> $\overline{x}$로 향하는 경로가 존재하고, 또한 $\overline{x}$ -> x로 향하는 경로가 존재할 경우. 즉 x와 $\overline{x}$를 포함하는 Strongly Connected Component가 존재할 경우.

이 경우에서 unsatisfiable함을 보이려면, 참->거짓인 edge가 항상 존재할 수밖에 없음을 보여야 한다.
먼저 v에 참을 할당해보자. 그 경우 v 뒤의 vertex들이 satisfiable을 만족하기 위해 모두 참이 되어야 한다. 하지만 v가 참이므로 $\overline{v}$는 거짓이 되고, 어느 한 edge에서는 결국 참->거짓의

경우가 생기게 되므로 이 경우는 항상 unsatisfiable하다.
v에 거짓을 할당해보자. 이 경우에는 $\overline{v}$가 참이므로 $\overline{v}$ 뒤의 vertex들은 모두 참으로 할당되어야 하는데, v가 거짓이기 때문에 어느 한 edge에서는 참->거짓의 경우가 생길 수밖에 없다. 따라서 이 경우도 unsatisfiable하다.
위의 두 경우에 따라서 항상 unsatisfiable하다.

임의의 literal과 그 complement를 포함하는 Strongly Connected Component가 존재하지 않는 경우에는 다음과 같이 할당해줄 수 있다.

모든 literal들이 할당될 때까지 다음 과정을 반복한다.
1. 임의의 literal x를 선택한다. 만약 x -*> $\overline{x}$의 경로가 존재하면 $\overline{x}$에 참을 할당하고, x에 거짓을 할당한다. 반대로 $\overline{x}$ -*> x의 경로가 존재하는 경우 x에 참을 할당하고 $\overline{x}$에 거짓을 할당한다. 이 때 참을 할당한 literal을 α라고 하자.(즉 $\overline{\alpha}$->α 존재)
2. 그 뒤, α에서 도달 가능한 모든 literal들에(β라고 하자) 대해 true를 할당한다. 그리고 $\overline{\beta}$에는 false를 할당한다.

이 때 모든 β에 true를 할당해도 괜찮은데, 그 이유는 만약 α에서 $\overline{\beta}$에 이르는 경로가 존재한다면(true->false가 되어 unsatisfiable하게 되는 경우), α -> β, a -> $\overline{\beta}$의 두 경로가 존재한다는 뜻이고 그렇다면 $\overline{\beta}$ -> $\overline{\alpha}$, β -> $\overline{\alpha}$의 경로가 따라서 존재하게 되고 결국 α->$\overline{\alpha}$의 경로가 존재하게 되어 임의의 literal과 그 complement를 포함하는 Strongly Connected Component가 존재하지 않는다는 조건에 모순이 생기게 된다. 즉 α에서 $\overline{\beta}$에 이르는 경로는 존재하지 않는다.
따라서 임의의 literal과 그 complement를 포함하는 Strongly Connected Component가 존재하지 않는 경우에는 위와 같은 알고리즘으로 할당하면 항상 satisfiable하게 할당할 수 있음을 알 수 있다.

위의 증명에 의해 2SAT문제는 Strongly Connected Component를 찾아서, 한 Strongly Connected Component에서 임의의 literal x에 대해 x와 $\overline{x}$가 모두 존재하는지를 찾는 문제와 같다. 그리고 이는 다항식 시간에 답을 찾을 수 있으므로 2SAT $\in$ P이다.

3. HAM-PATH 문제는 아래와 같이 정의된다.

Input: a graph G=(V, E), two vertices u and v
Question: Is there a Hamiltonian path from u and v ?
Definition: Hamiltonian path- a simple path that visits every vertex exactly once

HAM-CYCLE이 NP-Complete임을 이용하여 HAM-PATH 문제가 NP-Hard임을 증명하여라.

HAM-CYCLE의 input G=(V, E)가 있을 때, 임의의 vertex u에 대해 u'를 만들고 u에 연결된 모든 edge 또한 u'에 연결한 그래프를 G'이라 하고 HAM-PATH의 input으로 G'와 (u, u')을 준다. 이는 다음과 같이 나타낼 수 있다.

HAM-CYCLE(G) -> HAM-PATH(G', u, u')

이 때 하나의 vertex에 edge가 최대 v-1개 연결될 수 있으므로 새로운 u'를 만들어 연결시키는 변환은 O(V) 정도의 다항식 시간 내에 가능하다.

(=>) G가 HAM-CYCLE을 만족할 때, G'에서 u에 연결된 모든 edge들이 u'에도 연결되어 있고, (u, u') edge는 없으므로 G'에서 u와 u'을 시작점과 끝점으로 하는 HAM-PATH가 존재한다.
(<=) (G', u, u')이 HAM-PATH라면, u'에 연결된 edge들은 모두 u에도 연결되어 있으므로 G도 HAM-CYCLE이다.

예외적으로 G의 vertex가 2개뿐이라면(예를 들어, u와 v) G가 항상 HAM-CYCLE이 될 수 없는데, 이 때는 edge (u,v)를 끊어 G'로 주면 G'도 항상 HAM-PATH가 되지 못하게 변환할 수 있다.

따라서 HAM-CYCLE <=p HAM-PATH이고, HAM-PATH는 NP-hard이다.

4. CLIQUE이 NP-Complete임을 이용하여 아래의 SUBGRAPH-ISOMORPHISM 문제가 NP-Complete임을 증명하라.

SUBGRAPH-·ISOMORPHISM

Input: Two graphs, $G=(V_1,E_1)$ and $H=(V_2, E_2)$.

Question: Does G contain a subgraph isomorphic to H, that is, a subset $V \subseteq V_1$ and a subset $E \subseteq E_1$ s.t. $|V|=|V_2|$, $|E|=|E_2|$, and there exists a 1-to-1 function $f: V_2 \rightarrow V$ satisfying $\{u,v\} \in E_2$ iff $\{f(u), f(v)\} \in E$?

Hint: 이것은 아주 쉬운 문제로 여러분이 NP-Complete의 증명 process를 가장 단순하게 연습할 수 있도록 하는 것이 목적임.

1. H와 f가 주어졌을 때, 모든 edge에 대해 맵핑을 통해 검사하면 되는데 따라서 O(E) 정도의 시간이 걸리며 다항식 시간 안에 확인할 수 있으므로 NP이다.

2. CLIQUE의 input G와 k가 있을 때, G는 SUBGRAPH-ISOMORPHISM의 G로 주고 k개짜리 vertex를 가지는 완전 그래프를 만든 뒤 그 그래프를 H로 준다. 즉
   CLIQUE(G, k) -> SUBGRAPH_ISOMORPHISM(G, completeGraph(k))
   이 때 완전 그래프는 k(k-1)/2개의 edge를 가지므로 만드는 데 $\Theta(k^2)$의 시간이 걸린다.
   따라서 다항식 시간의 복잡도로 변환할 수 있다.
   이렇게 변환했을 때, G가 주어진 H와 동형인 그래프를 가진다는 것은 G가 크기 k의 완전부분그래프를 가진다는 것과 동일한 의미를 가진다.
   따라서 CLIQUE <=p SUBGRAPH-ISOMORPHISM 이므로
   SUBGRAPH-ISOMORPHISM은 NP-hard이다.

1, 2에 의해 SUBGRAPH-ISOMORPHISM 문제는 NP이고 NP-hard이므로 NP-complete이다.

5. 문제 PARTITION은 다음과 같이 정의된다.

Input: a finite set A and a size $s(a) \in Z^+$ for each $a \in A$.

Question: Is there a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$$

PARTITION이 NP-complete란 사실을 이용하여 아래의 SCHEDULING 문제가 NP-Complete임을 증명하라.

SCHEDULING

Input: A finite set T of tasks and, for each $t \in T$, an integer release time $r(t) \geq 0$, a deadline $d(t) \in Z^+$, and a length $l(t) \in Z^+$.

Question: Is there a feasible schedule for T, that is, a function $\sigma: T \rightarrow Z^+$ such that, for each $t \in T$, $\sigma(t) \geq r(t)$, $\sigma(t)+l(t) \leq d(t)$, and, for a, $b \in T$ $(a \neq b)$, either $\sigma(a)+l(a) \leq \sigma(b)$ or $\sigma(b)+l(b) \leq \sigma(a)$?

Hint on transformation: PARTITION의 element 하나당 SCHEDULING의 task 하나를 대응시키되 추가로 하나의 task를 추가하여 그것이 맨 가운데 실행되도록 강요하여, partition이 제대로 안되면 스케쥴링도 제대로 안되도록 만든다.

1. task 집합과 function σ이 주어졌을 때 각각 task마다 주어진 조건을 만족하는지 확인하면 되므로 확인에는 $\Theta(T^2)$의 시간이 걸린다. 따라서 다항식 시간에 확인할 수 있으므로 NP이다.

2. PARTITION의 input으로 집합 A와 원소 a들의 size s(a)가 주어지면, PARTITION의 집합 A의 원소 a에 각각 하나씩 task t를 할당하고, 그 size를 l(t)로 할당한다. 모든 task에서 release time r(t) = 0, deadline $d(t) = 1 + \sum_{t \in T} l(t)$ 로 동일하게 할당해준다. 그리고 하나의

task t_m을 생성해서 $l(t_m)$ = 1, $r(t_m)$ = (1+$\sum_{t\in T} l(t)$) / 2, d(t_m) = r(t_m) + 1로 두어 항상 맨 가운데 실행될 수 있도록 한다. 연산 '/'는 몫을 뜻하는데, 예를 들어 4/2 = 2이고 5/2 = 2이다.
이 때 size의 합, 즉 $\sum_{t\in T} l(t)$가 홀수라면, PARTITION이 항상 불가능하므로 SCHEDULING도 항상 불가능하게 변환해주자. 예를 들어 r(t) > d(t)가 되도록 할당해줄 수 있다.

이제 PARTITION에서 "두 부분 집합의 합이 같게 되도록 하는 A의 두 부분 집합이 있는가?"라는 질문과 변환된 SCHEDULING의 "요구조건을 만족하는 스케쥴링이 가능한가?"가 같음을 보이면 된다.

증명 과정에 1+$\sum_{t\in T} l(t)$의 값이 자주 사용되는데, (1+$\sum_{t\in T} l(t)$)/2 = k로 둔다. 이 때 1+$\sum_{t\in T} l(t)$는 변환에 의해 항상 홀수이므로 1+$\sum_{t\in T} l(t)$ = 2k + 1이 된다.

(=>) PARTITION에서 두 부분집합의 합이 같은 A', A"가 있다고 하자. 그러면 우선 A'에 할당된 task들부터 임의의 순서로 수행할 수 있다. 이 A'의 task들은 정확히 k의 시간이 되면 끝날 것이다. 그리고 이 시간부터 t_m을 1의 시간동안 수행한 뒤 A"에 할당된 task들을 임의의 순서로 수행한다. 2k+1 = 1+$\sum_{t\in T} l(t)$의 시간이 되면 모든 작업이 끝나게 된다. 따라서 SCHEDULING을 만족한다.

(<=) 요구 조건을 만족하는 스케쥴링이 가능하다고 하자. 이 때 k의 시간부터 k + 1의 구간에 t_m이 수행되므로, T에 속하는 나머지 task들은 0 ~ k의 구간과 k + 1 ~ 2k + 1의 구간에 나누어지게 된다. 두 구간을 A'와 A"라고 할 때, A'의 원소들의 합과 A"의 원소들의 합은 k - 0 = 2k + 1 - (k + 1) = k로 같다. 따라서 PARTITION에서 두 부분집합의 합이 같도록 하는 A의 두 부분집합이 존재한다.

위의 두 과정에 의해 PARTITION <=p SCHEDULING이므로 SCHEDULING 문제는 NP-hard이다.

1,2에 의해 SCHEDULING은 NP이고 NP-hard이므로 NP-complete이다.

6. Triangle inequality를 만족하지 않는 TSP의 경우에는 P=NP가 아닌 한 어떠한 상수 ρ에 대해서도 최적해의 ρ배를 넘지 않는 해를 보장할 수 있는 polynomial-time 알고리즘이 존재하지 않는다는 사실을 배웠다. 이제 다음의 경우를 생각해 보자.

도시들 간의 거리가 '1부터 100까지의 정수' 값 중에 임의로 정해진 TSP가 있다. 즉, 삼각부등식을 만족하지 않는다. 삼각부등식을 만족하지 않지만 도시간의 거리가 일정한 상수 범위 안에 있다. 이러한 성질을 만족하는 TSP에 대해서, 여러분이 배운 TSP를 위한 근사 알고리즘을 이용하여 어느 정도까지 품질을 보장할 수 있는 방법이 있다. 이 방법을 밝히고 보장할 수 있는 품질은 어느 정도인지 이야기하라.

이 문제의 경우 삼각부등식을 만족하게 할 수 있다. 도시들 간의 거리가 1~100이므로 삼각부등식을 만족하지 않는 최악의 경우는 100, 1, 1이라는 경우이다.

이 경우 삼각부등식을 만족시키려면 100+x <= 2*(1+x)를 만족하면 되므로 x >= 98, 즉 모든 거리에 98 이상의 값을 더해주면 항상 삼각부등식을 만족시킬 수 있다. 그 중 최솟값인 98을 더해주자.

최적해 C가 있을 때, 모든 edge의 weight에 98씩 더해주었으므로 이 경우의 최적해는 C+98V(V: vertex 개수)가 된다. 삼각부등식을 만족하는 경우 최적해의 3/2배를 넘지 않는 근사해를 구할 수 있으므로, 이 경우 3/2(C+98V) = C+1/2(C+98V)+98V의 품질을 보장할 수 있다. 이 때 기존에 더해주었던 98V를 다시 빼주면, C+1/2(C+98V)가 된다.

즉 최적해 C보다 1/2(C+98V) 이상 크지 않은 근사해의 품질을 보장할 수 있다.