Полиномиальный в среднем алгоритм для «SAT»

Н. Н. Кузюрин С. А. Фомин

18 февраля 2011 г.

Алгоритм динамического программирования для задачи «SAT» («Выполнимость»). Полиномиальность алгоритма «в среднем».

Полиномиальность в среднем

Определение

«Полиномиальный в среднем (точно)»
Алгоритм A называется ***полиномиальным в среднем****, если среднее время его работы ограничено полиномом от длины входа, т. е.*

$$\exists C > 0 : \quad \mathbf{E}_n \, T_A = O(n^C).$$

Полиномиальность в среднем

Определение

«Полиномиальный в среднем (точно)»
Алгоритм A называется ***полиномиальным в среднем****, если среднее время его работы ограничено полиномом от длины входа, т. е.*

$$\exists C > 0 : \quad \mathbf{E}_n T_A = O(n^C).$$

Упражнение

Приведите пример функции T_A (времени работы некоторого алгоритма A) и распределения исходных данных $\mathsf{P}_n(I)$, для которых T_A является полиномиальной в среднем ($\mathbf{E}_n T_A = O(n^c)$), а T_A^2 — нет.

Полиномиальность в среднем

Определение

«Полиномиальный в среднем (точно)»
*Алгоритм A называется **полиномиальным в среднем**, если среднее время его работы ограничено полиномом от длины входа, т. е.*

$$\exists C > 0 : \quad \mathbf{E}_n T_A = O(n^C).$$

Упражнение

Приведите пример функции T_A (времени работы некоторого алгоритма A) и распределения исходных данных $\mathrm{P}_n(I)$, для которых T_A является полиномиальной в среднем ($\mathbf{E}_n T_A = O(n^c)$), а T_A^2 — нет.

Определение

«Полиномиальный в среднем»
*Алгоритм называется **полиномиальным в среднем**, если для времени работы алгоритма T выполняется:*

Задача «SAT» («Выполнимость»)

Задача

«Выполнимость/SAT»[a]. Дано булевское выражение, являющееся ***конъюнктивной нормальной формой*** *(КНФ):*

$$CNF = \bigwedge_{i=1}^{m} C_i, \tag{1}$$

где C_i — элементарные дизъюнкции вида

$$x_{j_1}^{\sigma_1} \vee \ \ldots \vee x_{j_k}^{\sigma_k}, \tag{2}$$

$1 \leq k \leq n$, $\sigma_j \in \{0, 1\}$, $x^1 = x$ *и* $x^0 = (\neg x)$.
Существует ли (булевский) набор переменных x_j, обращающий эту форму в 1 *(т. е. в «Истину»)?*

---

[a] В англоязычной литературе — Satisfiability или просто SAT.

Как решать?

1. Перебор всех входных наборов $x = \{x_1, \ldots, x_n\}$, пока $CNF(x) = 0$. $\Rightarrow$ В худшем случае (CNF невыполнима) надо перебрать 2^n наборов x.

Как решать?

1. Перебор всех входных наборов $x = \{x_1, \ldots, x_n\}$, пока $CNF(x) = 0$. $\Rightarrow$ В худшем случае (CNF невыполнима) надо перебрать 2^n наборов x.
2. Подсчитать количество невыполнимых наборов — $|X_0|$, $X_0 = \{x : CNF(x) = 0\}$. Если $|X_0| = 2^n \Leftrightarrow CNF$ — невыполнима.

Как решать?

1. Перебор всех входных наборов $x = \{x_1, \ldots, x_n\}$, пока $CNF(x) = 0$. $\Rightarrow$ В худшем случае (CNF невыполнима) надо перебрать 2^n наборов x.
2. Подсчитать количество невыполнимых наборов — $|X_0|$, $X_0 = \{x : CNF(x) = 0\}$. Если $|X_0| = 2^n \Leftrightarrow CNF$ — невыполнима.

Как подсчитать $|X_0|$, более эффективно, нежели перебором x?

Формула включений-исключений

$CNF(x) = 0 \Rightarrow$ одна или несколько скобок-дизъюнкций $C_j(x) = 0$.
Для подсчета мощности объединения пересекающихся множеств применяется комбинаторная формула включений-исключений

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n} |A_i| - \sum_{i,j\,:\,i<j} |A_i \cap A_j| + \\ + \sum_{i,j,k\,:\,i<j<k} |A_i \cap A_j \cap A_k| - \ \cdots\cdots\ (-1)^{n-1} |A_1 \cap \cdots \cap A_n| \,.$$

В нашем случае A_i обозначает множество наборов, для которых i-я скобка C_i равна нулю, поэтому

$$|X_0| = |\cup_{i=1}^{m} A_i|.$$

Скобки, литералы, покрытие

Определение

Литерал *— каждое вхождение переменной x_i (или ее отрицания) в скобку. Например, для $(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_3)$ литералами будут $x_1, \overline{x_2}, \overline{x_1}, x_3$.*

Рассмотрим некоторое подмножество из k скобок $S = \{C_{j_1}, \ \ldots, C_{j_k}\}$.

Lit_S — литералы S, все литералы $lit = \{x_i, \overline{x_i}\}$, входящие в S.

Cov_S — покрытие S, все переменные x_i, входящие в S.

Зависимые и независимые множества скобок.

$CNF(x) = 0 \Rightarrow$ одна или несколько скобок-дизъюнкций $C_j(x) = 0$.

Зависимые и независимые множества скобок.

$CNF(x) = 0 \Rightarrow$ одна или несколько скобок-дизъюнкций $C_j(x) = 0$.
Рассмотрим некоторое подмножество из k скобок $S = \{C_{j_1}, \ \ldots, C_{j_k}\}$.
Когда все скобки в S будут равны нулю?

Зависимые и независимые множества скобок.

$CNF(x) = 0 \Rightarrow$ одна или несколько скобок-дизъюнкций $C_j(x) = 0$.
Рассмотрим некоторое подмножество из k скобок $S = \{C_{j_1}, \ \ldots, C_{j_k}\}$.
Когда все скобки в S будут равны нулю?
$\Rightarrow Z_S(x) = 1$, где

$$Z_S(x) = \bigwedge_{i=1}^{k} \overline{C_{j_i}} = \bigwedge_{i=1}^{k} \overline{\bigvee_{lit \in C_{j_i}} lit} = \bigwedge_{i=1}^{k} \bigwedge_{lit \in C_{j_i}} \overline{lit} = \bigwedge_{lit \in Lit_S} \overline{lit}$$

Зависимые и независимые множества скобок.

$CNF(x) = 0 \Rightarrow$ одна или несколько скобок-дизъюнкций $C_j(x) = 0$.
Рассмотрим некоторое подмножество из k скобок $S = \{C_{j_1}, \ \ldots, C_{j_k}\}$.
Когда все скобки в S будут равны нулю?
$\Rightarrow Z_S(x) = 1$, где

$$Z_S(x) = \bigwedge_{i=1}^{k} \overline{C_{j_i}} = \bigwedge_{i=1}^{k} \overline{\bigvee_{lit \in C_{j_i}} lit} = \bigwedge_{i=1}^{k} \bigwedge_{lit \in C_{j_i}} \overline{lit} = \bigwedge_{lit \in Lit_S} \overline{lit}$$

т. е. $Z_S(x) = 1 \Leftrightarrow \forall lit \in S \; lit(x) = 0$.

Зависимые и независимые множества скобок.

$CNF(x) = 0 \Rightarrow$ одна или несколько скобок-дизъюнкций $C_j(x) = 0$.
Рассмотрим некоторое подмножество из k скобок $S = \{C_{j_1}, \ \ldots, C_{j_k}\}$.
Когда все скобки в S будут равны нулю?
$\Rightarrow Z_S(x) = 1$, где

$$Z_S(x) = \bigwedge_{i=1}^{k} \overline{C_{j_i}} = \bigwedge_{i=1}^{k} \overline{\bigvee_{lit \in C_{j_i}} lit} = \bigwedge_{i=1}^{k} \bigwedge_{lit \in C_{j_i}} \overline{lit} = \bigwedge_{lit \in Lit_S} \overline{lit}$$

т. е. $Z_S(x) = 1 \Leftrightarrow \forall lit \in S \ lit(x) = 0$.
Множество S может быть:

зависимое — $\exists i$: литералы $x_i \in S$ и $\overline{x_i} \in S \Rightarrow Z_S \equiv 0$.
$|x : Z_S(x) = 1| = 0$.

независимое — $\nexists i$: $x_i \in S$ и $\overline{x_i} \in S$. Следовательно, $Z_S(x) = 1 \Rightarrow$
$\forall x_i \in Cov_S$ значение определено.
$|x : Z_S(x) = 1| = 2^{n-|Cov_S|} = 2^{n-|Lit_S|}$.

Подсчет «нулевых» наборов

$CNF(x) = 0 \Rightarrow \exists$ независимое S, $\Rightarrow$ построим все независимые S,
и для них посчитаем число «обнуляющих» наборов $2^{n-|Lit_S|}$.

Подсчет «нулевых» наборов

$CNF(x) = 0 \Rightarrow \exists$ независимое S, $\Rightarrow$ построим все независимые S,
и для них посчитаем число «обнуляющих» наборов $2^{n-|Lit_S|}$.
Как суммировать?

Подсчет «нулевых» наборов

$CNF(x) = 0 \Rightarrow \exists$ независимое S, $\Rightarrow$ построим все независимые S,
и для них посчитаем число «обнуляющих» наборов $2^{n-|Lit_S|}$.
Как суммировать?

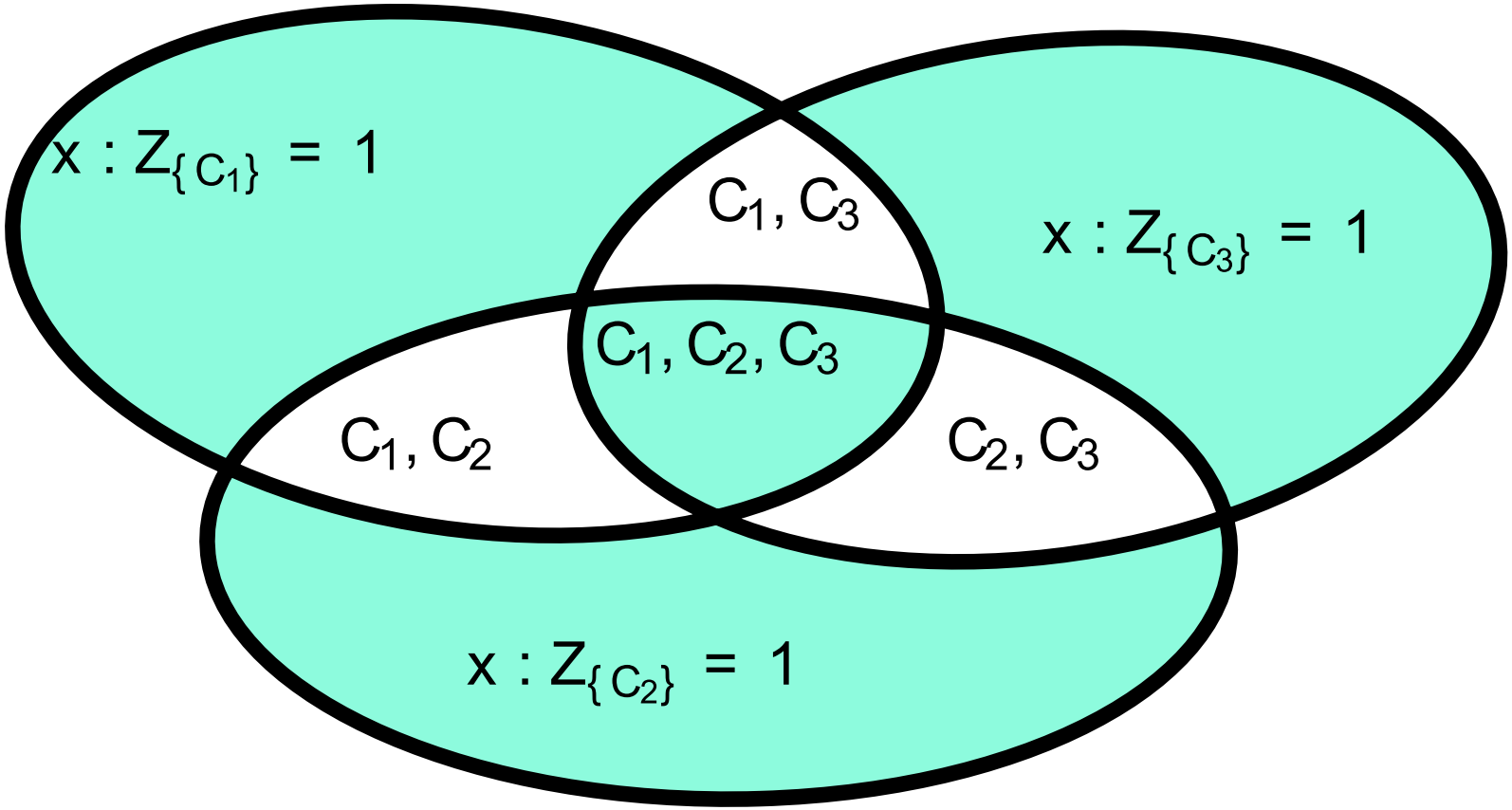
Лемма

Любое подмножество независимого множества — независимо.

Далее удобно разбить все независимые множества на «k-слои» N_k: независимые множества с k скобками, и организовать суммирование включений-исключений по этим слоям:

$$|X_0| = \sum_{k=1}^{m} (-1)^{k-1} \cdot \sum_{S \in N_k} 2^{n-|Lit_S|}.$$

Формула «включений/исключений»



Формула «включений/исключений»:

$$|X_0| = \Sigma_{k=1}^{m} \Sigma_{S \in N_k} (-1)^k \cdot 2^{n-|Lit_S|}.$$

Построение независимых множеств

N_{k+1} строится из N_k

for all $C_j \in \{C_1, \ldots, C_m\}$ **do**
  for all $S \in N_k$ **do**
    $Lit_{S'} \leftarrow Lit_S$
    for all $lit \in C_j$ **do**
      if $\overline{lit} \notin Lit_{S'}$ **then**
        $Lit_{S'} \leftarrow Lit_{S'} \cup lit$
      else
        отбрасываем зависимое $C_j + S$.
      end if
    end for
  end for
end for

```
def SatDynP(cnf):
    size = 0                          #счетчик выполняющих КНФ наборов
    n = cnf.numVariables()  #число переменных в КНФ
    T = {cnf.getEmptySetOfClauses(): SetOfLiterals([])}
    for k, dummy in enumerate(cnf): # цикл по слоям
        S = {}          # новый слой независимых множеств
        for j, C in enumerate(cnf):   # цикл по скобкам
            for t in T:             # цикл по независимым множествам
                if j not in t:    # если скобки в множестве нет
                    s = t + j          # добавляем её к множеству
                    if s not in S:    #в слое S нет такого множесва?
                        lit = copy(T[t]) #наследуем литералы из t
                        for x in C:     # для каждого литерала в C
                            ok = -x not in lit  # x совместен с другими?
                            if not ok:
                                break
                            lit.add(x)
                    if ok and len(lit) > len(T[t]):
                        S[s] = lit    # да, s – независимое
                        size += 2**(n - len(lit)) * (-1)**k
        T = S
    return size
```

$$CNF = (x_2 \vee x_3 \vee \bar{x}_1) \wedge (x_3 \vee \bar{x}_2) \wedge (x_1 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee x_4)$$

k	*size*	*T*
0	0	$\{\}$: $\{\}$
1	12	$\{C_0\}$: $\{x_2, x_3, \bar{x}_1\}$, $\{C_3\}$: $\{x_1, x_2, x_4\}$, $\{C_1\}$: $\{x_3, \bar{x}_2\}$, $\{C_2\}$: $\{x_1, \bar{x}_3\}$
2	11	$\{C_2, C_3\}$: $\{x_1, x_2, x_4, \bar{x}_3\}$
3	11	
end	11	

$11 < 2^4 \rightarrow$ КНФ выполнима.

$$CNF = (x_3 \vee \bar{x}_2) \wedge (x_1 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_1) \wedge (x_2 \vee x_3)$$

k	*size*	*T*
0	0	$\{\}$: $\{\}$
1	8	$\{C_0\}$: $\{x_3, \bar{x}_2\}$, $\{C_3\}$: $\{x_2, x_3\}$, $\{C_1\}$: $\{x_1, \bar{x}_3\}$, $\{C_2\}$: $\{\bar{x}_3, \bar{x}_1\}$
2	8	
3	8	
end	8	

$8 = 2^3 \rightarrow$ КНФ невыполнима.

Лемма

Сложность алгоритма в наихудшем случае — $O(m^2 n \cdot \max_k |N_k|)$.

Лемма

Сложность алгоритма в наихудшем случае — $O(m^2 n \cdot \max_k |N_k|)$.

Упражнение

Какие входные данные для этого алгоритма заставят его работать экспоненциально долго?

Лемма

Сложность алгоритма в наихудшем случае — $O(m^2 n \cdot \max_k |N_k|)$.

Упражнение

Какие входные данные для этого алгоритма заставят его работать экспоненциально долго?

Упражнение

На каких входных данных время работы этого алгоритма будет $O(m)$?

Лемма

Сложность алгоритма в наихудшем случае — $O(m^2 n \cdot \max_k |N_k|)$.

Упражнение

Какие входные данные для этого алгоритма заставят его работать экспоненциально долго?

Упражнение

На каких входных данных время работы этого алгоритма будет $O(m)$?

Теорема

Пусть для каждой скобки вероятность появления каждой из n переменных (или ее отрицания) равна p, причем

$$np^2 \geq \ln m.$$

Тогда алгоритм является полиномиальным в среднем.

Доказательство

S_k — некоторое множество скобок, $|S_k| = k$;

$p_i(S_k)$ — $\mathrm{P}[$в S_k нет одновременно x_i и $\neg x_i]$;

$P(S_k)$ — $\mathrm{P}[S_k$ — независимое$]$.

Доказательство

S_k — некоторое множество скобок, $|S_k| = k$;

$p_i(S_k)$ — P[в S_k нет одновременно x_i и $\neg x_i$];

$P(S_k)$ — P[S_k — независимое].

$$p_i(S_k) = (1-p)^k + (1-p)^k - (1-p)^{2k} = (1-p)^k(2-(1-p)^k) \leq$$
$$\leq (1-p)^k(1+kp) \leq (1-p)^k(1+p)^k = (1-p^2)^k.$$

Доказательство

S_k — некоторое множество скобок, $|S_k| = k$;

$p_i(S_k)$ — P[в S_k нет одновременно x_i и $\neg x_i$];

$P(S_k)$ — P[S_k — независимое].

$$p_i(S_k) = (1-p)^k + (1-p)^k - (1-p)^{2k} = (1-p)^k(2-(1-p)^k) \leq$$
$$\leq (1-p)^k(1+kp) \leq (1-p)^k(1+p)^k = (1-p^2)^k.$$

$$P(S_k) = \prod_{i=1}^{n} p_i(S_k) \leq (1-p^2)^{kn}.$$

Доказательство

S_k — некоторое множество скобок, $|S_k| = k$;

$p_i(S_k)$ — P[в S_k нет одновременно x_i и $\neg x_i$];

$P(S_k)$ — P[S_k — независимое].

$$p_i(S_k) = (1-p)^k + (1-p)^k - (1-p)^{2k} = (1-p)^k(2-(1-p)^k) \le$$
$$\le (1-p)^k(1+kp) \le (1-p)^k(1+p)^k = (1-p^2)^k.$$

$$P(S_k) = \prod_{i=1}^{n} p_i(S_k) \le (1-p^2)^{kn}.$$

$$\mathbf{E}\,|N_k| \le \sum_{k=1}^{m} \binom{m}{k} P(S_k) \le \sum_{k=1}^{m} \binom{m}{k} (1-p^2)^{kn} \le$$
$$\le \sum_{k=1}^{m} m^k \exp\{-np^2k\} \le \sum_{k=1}^{m} \exp\{k(\ln m - np^2)\} \le \sum_{k=1}^{m} 1 \le m.$$

«Карта памяти» лекции

Интернет поддержка курса

`http://discopal.ispras.ru/`

Вопросы?