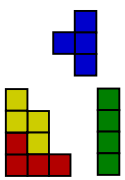
Полиномиальный в среднем алгоритм для «Упаковки»

Н. Н. Кузюрин С. А. Фомин

10 декабря 2010 г.



Алгоритм динамического программирования для задачи упаковки подмножеств. Полиномиальность алгоритма «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.*

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

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 (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.

Определение

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

$$\exists \varepsilon > 0 : \quad \mathbf{E}_n T^\epsilon = O(n).$$

Задача об упаковке

Задача об упаковке

Задача

«Упаковка» (Packing)
Дано конечное множество L из m элементов и система его подмножеств S_1, …, S_n. Требуется найти максимальную по числу подмножеств подсистему попарно непересекающихся подмножеств.

Задача об упаковке

Задача

«Упаковка» (Packing)
Дано конечное множество L из m элементов и система его подмножеств S_1, ..., S_n. Требуется найти максимальную по числу подмножеств подсистему попарно непересекающихся подмножеств.

Определение

Пусть

$L = \{l_1, \ldots, l_m\}$ — *m-элементное множество;*

$\{S_1, \ldots, S_n\}$ — *семейство подмножеств L;*

Тогда

- *Элемент l_i и подмножество S_j* ***инцидентны****, если $l_i \in S_j$.*
- ***Матрицей инцидентности*** *называется $\{0,1\}$-матрица $A = (a_{ij})$ размером $m \times n$, для которой:*
  $a_{ij} = 1 \Leftrightarrow$ *элемент l_i и подмножество S_j инцидентны.*

ЦЛП для «Упаковки»

$$\sum_{j=1}^{n} x_j \to \max$$

$$\forall j : \ 1 \le j \le n \quad x_j \in \{0, 1\}$$

$$\forall i : \ 1 \le i \le m \quad \sum_{j=1}^{n} a_{ij} x_j \le 1$$

ЦЛП для «Упаковки»

$$\sum_{j=1}^{n} x_j \to \max$$

$$\forall j: \ 1 \le j \le n \quad x_j \in \{0, 1\}$$

$$\forall i: \ 1 \le i \le m \quad \sum_{j=1}^{n} a_{ij} x_j \le 1$$

ЦЛП для «Упаковки» (общий случай)

$$\sum_{j=1}^{n} c_j x_j \to \max$$

$$\forall j: \ 1 \le j \le n \quad x_j \in \{0, 1\}$$

$$\forall i: \ 1 \le i \le m \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i$$

```python
def PackingDynP(A):

    m, n = A.shape
    X=[( zeros(n, int), #решение
         zeros(m, int), #покрытие
         0  )]          #размер покрытия
    for j in xrange(n):
        Sj = get_column(j)
        for sol, cov, size in X[:]:
            newcov = cov+Sj
            if max(newcov) <= 1:
                newsol = copy(sol)
                newsol[j] = 1
                X.append( (
                    newsol,
                    newcov,
                    size + 1) )
    return get_optimal_set(X)
```

```
[[0 1 0 1]
 [1 0 1 0]
 [0 1 1 0]] --> 3x4-matrix A

column(0)+< [0 0 0 0], [0 0 0]>
  => [0 1 0]+[0 0 0]=[0 1 0] => [1 0 0 0]
column(1)+< [0 0 0 0], [0 0 0]>
  => [1 0 1]+[0 0 0]=[1 0 1] => [0 1 0 0]
column(1)+< [1 0 0 0], [0 1 0]>
  => [1 0 1]+[0 1 0]=[1 1 1] => [1 1 0 0]
column(2)+< [0 0 0 0], [0 0 0]>
  => [0 1 1]+[0 0 0]=[0 1 1] => [0 0 1 0]
column(3)+< [0 0 0 0], [0 0 0]>
  => [1 0 0]+[0 0 0]=[1 0 0] => [0 0 0 1]
column(3)+< [1 0 0 0], [0 1 0]>
  => [1 0 0]+[0 1 0]=[1 1 0] => [1 0 0 1]
column(3)+< [0 0 1 0], [0 1 1]>
  => [1 0 0]+[0 1 1]=[1 1 1] => [0 0 1 1]
Выбран набор  [1 1 0 0]  размером 2
```

Упражнение

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

Упражнение

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

Упражнение

На каких входных данных этот алгоритм будет работать $O(n^3)$?

Упражнение

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

Упражнение

На каких входных данных этот алгоритм будет работать $O(n^3)$?

Теорема

Пусть a_{ij} являются независимыми случайными величинами, принимающими значения $\{0, 1\}$, причем выполняется:

$$\mathsf{P}\{a_{ij} = 1\} = p$$
$$\mathsf{P}\{a_{ij} = 0\} = 1 - p$$
$$mp^2 \geq \ln n.$$

Тогда алгоритм является полиномиальным в среднем.

Мат. ожидание времени работы: $O(nm\,\mathbf{E}\,|T(n)|)$.

Мат. ожидание времени работы: $O\left(nm\,\mathbf{E}\left|T(n)\right|\right)$.

$\bar{x}^k$ — вектор с k единицами и $n-k$ нулями;

p_{ki} — вероятность выполнения i-неравенства для $\bar{x}^k$.

P_k — вероятность того, что $\bar{x}^k$ — допустимое решение;

Мат. ожидание времени работы: $O(nm\,\mathbf{E}\,|T(n)|)$.

$\bar{x}^k$ — вектор с k единицами и $n-k$ нулями;

p_{ki} — вероятность выполнения i-неравенства для $\bar{x}^k$.

P_k — вероятность того, что $\bar{x}^k$ — допустимое решение;

$$p_{ki} \equiv \mathsf{P}\left\{\sum_{j=1}^{n} a_{ij}x_j^k \le 1\right\} \le \mathsf{P}\left\{\sum_{j=1}^{k} a_{ij} \le 1\right\} =$$

$$= (1-p)^k + kp(1-p)^{k-1} = (1-p)^{k-1}(1+p(k-1)) \le$$

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

Мат. ожидание времени работы: $O(nm\,\mathbf{E}\,|T(n)|)$.

$\bar{x}^k$ — вектор с k единицами и $n-k$ нулями;

p_{ki} — вероятность выполнения i-неравенства для $\bar{x}^k$.

P_k — вероятность того, что $\bar{x}^k$ — допустимое решение;

$$p_{ki} \equiv \mathsf{P}\left\{\sum_{j=1}^{n} a_{ij}x_j^k \le 1\right\} \le \mathsf{P}\left\{\sum_{j=1}^{k} a_{ij} \le 1\right\} =$$

$$= (1-p)^k + kp(1-p)^{k-1} = (1-p)^{k-1}(1+p(k-1)) \le$$

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

$$\mathsf{P}_k = \prod_{i=1}^{m} p_{ki} \le \prod_{i=1}^{m} e^{-p^2(k-1)} = e^{-mp^2(k-1)}.$$

$$\mathbf{E}\,|T(n)| \le \sum_{k=0}^{n} \binom{n}{k} \mathsf{P}_k = 1 + \sum_{k=1}^{n} \binom{n}{k} \mathsf{P}_k \le 1 + n + \sum_{k=2}^{n} \binom{n}{k} \mathsf{P}_k <$$

$$< 1 + n + \sum_{k=2}^{n} \binom{n}{k} e^{-mp^2(k-1)} \le 1 + n + \sum_{k=2}^{n} e^{k \ln n - mp^2(k-1)} =$$

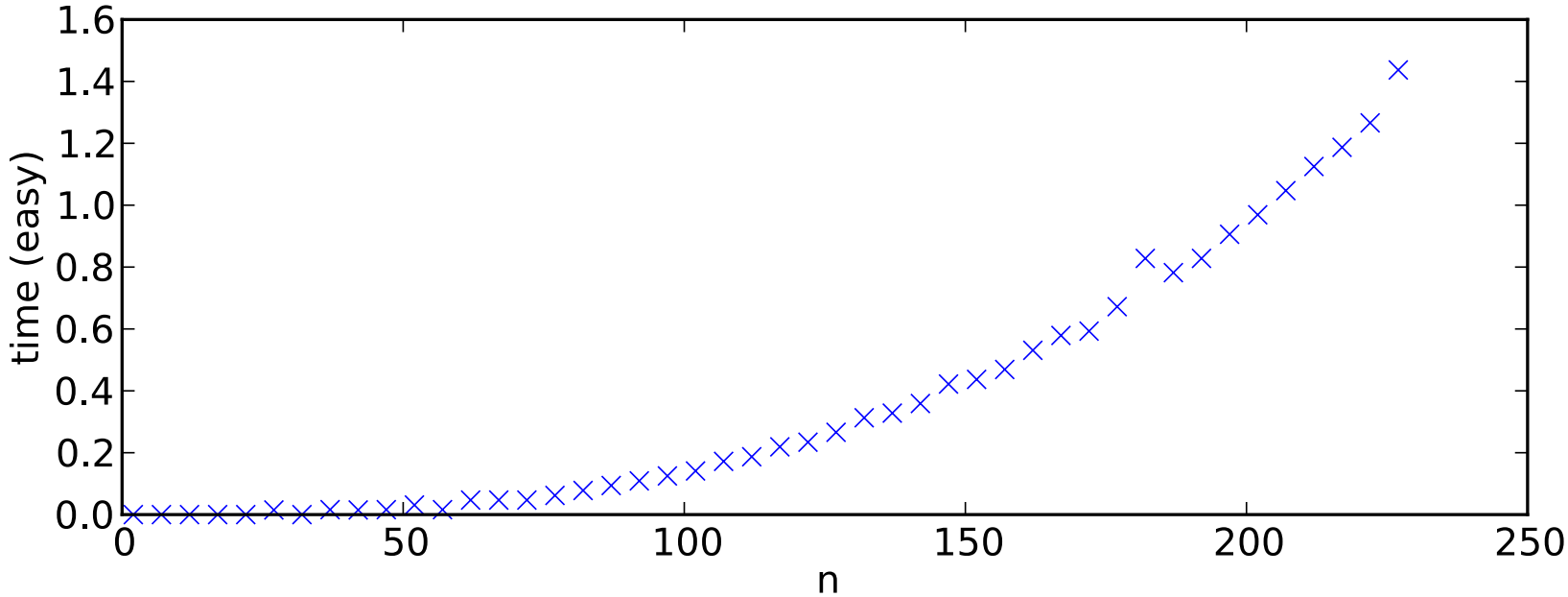
$$= 1 + n + n \sum_{k=2}^{n} e^{(k-1)(\ln n - mp^2)}.$$

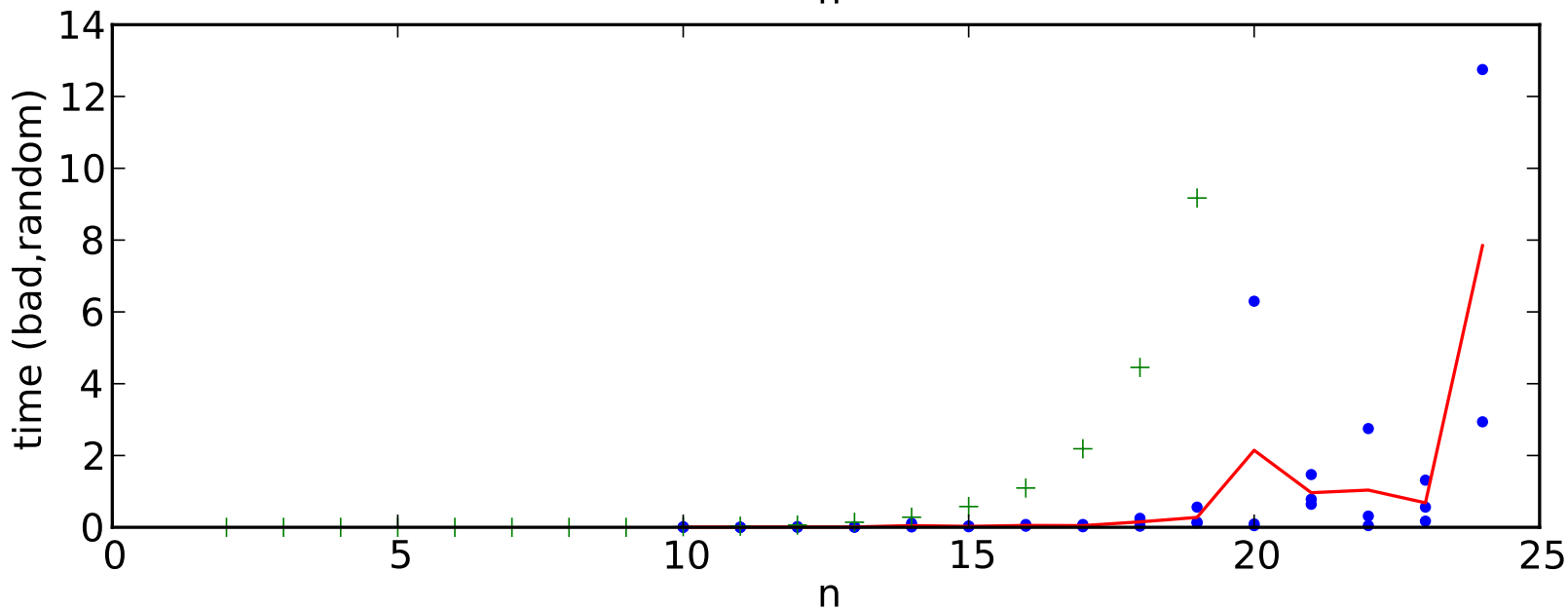
$$\mathbf{E}\,|T(n)| \le \sum_{k=0}^{n} \binom{n}{k} \mathsf{P}_k = 1 + \sum_{k=1}^{n} \binom{n}{k} \mathsf{P}_k \le 1 + n + \sum_{k=2}^{n} \binom{n}{k} \mathsf{P}_k <$$

$$< 1 + n + \sum_{k=2}^{n} \binom{n}{k} e^{-mp^2(k-1)} \le 1 + n + \sum_{k=2}^{n} e^{k \ln n - mp^2(k-1)} =$$

$$= 1 + n + n \sum_{k=2}^{n} e^{(k-1)(\ln n - mp^2)}.$$

При условии $mp^2 \ge \ln n$ в последней сумме каждый член не превосходит 1.
Это и означает, что $\mathbf{E}[T(n)] = O(n^2)$.


time (easy)
n
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
50
100
150
200
250


time (bad,random)
n
0
2
4
6
8
10
12
14
0
5
10
15
20
25

Интернет поддержка курса

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

Вопросы?