Оценка сложности вычислительных программ

лекция 22

План лекции

- ВременнАя и ёмкостная сложность программы
  - Программа с точки зрения сложности
  - Размер входных данных
  - Сложность в худшем, в среднем
- Понятие оптимальной программы
- Классы вычислительной сложности программ
  - Эквивалентность по сложности
  - Примеры классов вычислительной сложности

Программа, размер входных данных

- Обозначим Ct(A, x) и Cs(A, x) «затраты» по времени и по памяти на вычисление результата для данного x с помощью программы A
- Обозначим |x| «размер» входных данных программы
  - |x| >= 0
  - Конкретный выбор |.| зависит от программы

Примеры

- Умножение матриц MM
  - $|x|$ = порядок матрицы x
  - $Cs(MM, x) = 3*|x|^2$
  - $Ct(MM, x)$ = число умножений = $|x|^3$
- Проверка на простоту пробными делениями TD
  - $|x| = x$
  - $Cs(TD, x) = |x|$
  - $1 <= Ct(TD, x)$ = число делений $<= \mathrm{sqrt}(|x|)-1$
- Сортировка простыми вставками I
  - $|x|$ = длина массива x
  - $Cs(I, x) = |x|$
  - $|x| -1 <= Ct(I, x)$ = число сравнений $<= |x| *(|x| -1)/2$
- Как еще можно определить размер входа и «затраты» для этих программ?

Временная сложность

- ВременнОй сложностью (сложностью по времени в худшем случае) программы A называется функция от размера входных данных $T(A, n) = \max\{ Ct(A, x) \mid |x|=n \}$

Пространственная сложность

- Пространственной сложностью (сложностью по памяти в худшем случае) программы A называется функция от размера входных данных $S(A, n) = \max\{ Cs(A, x) \mid |x|=n \}$

Пример – временная сложность TD

- Пусть |x| = число битов в x

|x|	2	3	4	5	6	7
x	2-3	4-7	8-15	16-31	32-63	64-127
n*	3	5	13	31	59	127
T(TD,|x|)	1	1	2	4	6	10

- Пусть |x| = x

|x|	111	112	113	114	115	116
T(TD,|x|)	2	1	9	1	4	1

Пример – возведение в степень

- Возведение в степень методом повторных квадратов RS
- Пусть в 2 с.с. x = β[k]β[k-1] … β[1]β[0], β[i]∈ {0,1}
- RS

```
q := a; u := 1;
for i = 0 to k do
     if β[i] = 1 then u := u * q fi;
     if i < k then q := q^2 fi;
od
```

- Чему равна временная сложность RS для |x| = x и для |x| = число битов в записи x?

Сложность в среднем 1/3

- Обозначим $In = \{ x \mid |x| = n \}$ множество входных данных размера n
- Обозначим $Pn(x)$ вероятность входных данных $x \in In$
  - Можно считать $Pn(x) = 1/($число элементов в $In)$
  - Иногда считают, что вероятность разных входных данных разная
- По определению вероятности $\Sigma_{x \in In} Pn(x) = 1$

Сложность в среднем 2/3

- Величина $\underline{T}(A, n) = \Sigma_{x \in In} Ct(A,x)Pn(x)$ называется временной сложностью программы A в среднем

Сложность в среднем 2/3

- Величина $\underline{S}(A, n) = \Sigma_{x \in In} Cs(A,x)Pn(x)$ называется пространственной сложностью программы А в среднем

Связь сложности в худшем случае и в среднем

- Сложность в среднем не превосходит сложность в худшем случае
- $\underline{T}(A, n) = \Sigma_{x \in In} Ct(A,x)Pn(x) <=$
  $<= \Sigma_{x \in In} \max \{ Ct(A,x) \mid |x| = n \} Pn(x) =$
  $= T(A, n) \; \Sigma_{x \in In} Pn(x) = T(A, n)$

Пример* – сложность в среднем RS

- $In = \{ x \mid 2^{(n-1)} \le x < 2^n-1 \}$
- $|x|$ = число битов в x
- $Pn(x)$ = 1/(число элементов в In) = $1/2^{(n-1)}$
- $\underline{T}(RS, n) =$
  $= Pn(2^{(n-1)}) \sum_{x \in In}(|x|+(\text{число битов=1 в } x)-2) =$
  $= n-2+1+ Pn(2^{(n-1)}) \sum_{x \in In}(\text{число битов=1 в } x)$

Пример* – сложность в среднем RS

- $kC(n,k) = nC(n-1,k-1)$
- $\Sigma_{x \in In}$(число битов=1 в x) =
  $= \Sigma_{0<=k<=n-1} kC(n-1,k) = \Sigma_{1<=k<=n-1} (n-1)C(n-2,k-1)$
  $= (n-1) \Sigma_{0<=k<=n-2} C(n-2,k) = (n-1)2\hat{}(n-2)$
- $T(RS, n) = n-1+Pn(0)\, \Sigma_{x \in In}$(число битов=1 в x) $= n-1 + Pn(0)(n-1)2\hat{}(n-2) = 3(n-1)/2 <= 2n-2$

Полиномиальные программы

- Программа называется программой с полиномиально ограниченной сложностью, если ее сложность O(|x|^d)
- Программа называется полиномиальной, если ее сложность полиномиально ограничена

Оптимальные алгоритмы

- Пусть АА – класс программ
- Программа А* называется оптимальной в классе АА, если для любой программы А из АА и любого размера n входных данных T(A*, n) <= T(A, n)

Пример* min max -- 1/4

- Пусть AA – все программы для одновременного нахождения минимума и максимума в массиве
- Покажем, что сложность по числу сравнений оптимальной программы 3n/2-2 и приведем оптимальную программу

Пример * min max -- 2/4

- Каждый этап произвольной программы V, решающей эту задачу, характеризуется 4 множествами элементов массива (A,B,C,D)
  - A — множество элементов, не участвовавших в сравнениях
  - B — множество элементов, которые во всех сравнениях оказывались большими
  - C — множество элементов, которые во всех сравнениях оказывались меньшими
  - D — множество элементов, которые в некоторых сравнениях были больше, а в других — меньше
- Начальная ситуация $(n, 0, 0, 0)$, конечная — $(0, 1, 1, n - 2)$
- Пусть $\lambda(a,b,c) = 3a/2 + b + c - 2$, где a, b и c -- число элементов в A, B и C

Пример * min max -- 3/4

Сравнение	(a,b,c,d)	Изменение λ
AA	$(a-2, b+1, c+1, d)$	-1
AB	$(a-1, b, c+1, d)$ | $(a-1, b, c, d+1)$	-1/2 | -3/2
AC	$(a-1, b+1, c, d)$ | $(a-1, b, c, d+1)$	-1/2 | -3/2
AD	$(a-1, b+1, c, d)$ | $(a-1, b, c+1, d)$	-1/2 |-1/2
BB	$(a, b-1, c, d+1)$	-1
BC	$(a, b-1, c-1, d+2)$ | (a, b, c, d)	-2|0
BD	$(a, b-1, c, d+1)$ | (a, b, c, d)	-1|0
CC	$(a, b, c-1, d+1)$	-1
CD	$(a, b, c-1, d+1)$ | (a, b, c, d)	-1|0
DD	(a, b, c, d)	0

- Начинаем с $\lambda = 3n/2-2$,
- Заканчиваем $\lambda = 0$
- За шаг уменьшаем не более, чем на 1
  - Почему??
- Всего шагов не менее $3n/2-2$

Пример * min max -- 4/4

- Построим оптимальную программу
- Дан массив из n элементов $x1$,... , xn
- Образуем пары $x1$, $x2$; $x3$, $x4$; …
- В каждой паре найдём минимум и максимум за одно сравнение
- Пусть $m1$, $m2$, … – массив минимальных элементов пар размера $n/2$
- Пусть $M1$, $M2$, ... – массив максимальных элементов пар размера $n/2$
- Минимальный элемент исходного массива среди mi
- Максимальный элемент исходного массива среди Mi
- Если на первом шаге был непарный элемент (n — нечётное), то на него потребуется ещё два сравнения с найденными минимумом и максимумом
- В итоге на каждую пару тратится 3 сравнения

- Функции f и g называются функциями одного порядка, если найдутся такие c1 и c2, что для любого набора n значений аргументов f и g

  $$c1|g(n)| < |f(n)| < c2|g(n)|$$

- Обозначается f ~ g
- Функция f -- омега функции g, если найдется такая константа c, что $|f(n)| > c\,|g(n)|$ для всех n
- Обозначается $f(n) = \Omega(g(n))$

Асимптотически оптимальная программа

- Программа A* называется асимптотически оптимальной (оптимальной по порядку сложности) в классе AA, если $T(A^*, n) = \Omega(T(A, n))$ для любой другой программы A из AA

Асимптотически оптимальная программа

- Если A* и B* -- оптимальные программы в классе AA, то $T(A^*, n) = \Omega(T(B^*, n))$ и $T(B^*, n) = \Omega(T(A^*, n))$ и $T(A^*, n) \sim T(B^*, n)$
- Оптимальная асимптотическая сложность определена однозначно

Классы сложности задач

- Под «задачей» будем понимать набор из трех объектов:
  - функция P(.), которую требуется вычислить
  - функция измерения входных данных |.|
  - функция измерения числа операций T(.,.) в алгоритме вычисления функции P(.)

Классы сложности задач

- Задача P не сложнее Q, если для любой программы QA, решающей задачу Q, найдётся программа PA, решающая задачу P, такая что $T(PA, n) = O(T(QA, n))$
- Обозначение $P \le Q$
- Задачи P и Q, для которых одновременно верно $P \le Q$ и $Q \le P$, называются эквивалентными (по сложности)
- Обозначение P >< Q

Пример

- Рассмотрим следующие задачи:
  - M: умножение 2-х целых чисел a и b
  - D: деление целого a битовой длины ≤ 2m на целое b битовой длины m
  - S: возведение в квадрат целого a
  - R: обращение целого a
- Покажем, что M >< D >< S >< R

Пример

- Можно доказать, что для |x| = число битов в x сложность f(.) любого из этих алгоритмов
  - не убывает
  - $f(m) >= m$
  - $af(m) <= f(am) <= a^2f(m)$ для $a > 1$

Пример

- $M < S$
  - $ab = ((a+b)^2-a^2-b^2)/2$
  - $T(MA, m) = T(SA, m+1)+2T(SA,m)+O(m) = O(T(SA,m))$
- $S < R$
  - $a^2 = 1/(1/a-1/(a+1))-a$
  - $T(SA, m) = O(T(RA, c*m))$ – так как делить нужно в c раз более точно

Пример

- $R < M$
  - $x[i]=2*x[i-1]-a*x[i-1]^2$
  - Сходится к 1/a и $x[i-1]=1/a*(1-\varepsilon) \implies x[i]=1/a*(1-\varepsilon^2)$
    - Почему?
  - $T(RA, m) = O(T(MA,m))$
- $M >< S >< R$
- $D < M$
  - $a/b = a*(1/b)$
- $R < D$ -- очевидно

Заключение

- Задача, размер задачи как характеристика объема входных данных
- ВременнАя и ёмкостная сложность программы как функции размера задачи
  - Верхняя, нижняя и средняя оценки
- Классы вычислительной сложности алгоритмов
  - Эквивалентность по сложности
  - Примеры классов вычислительной сложности
- Примеры определения класса вычислительной сложности задач