Структури от данни
+
алгоритми
=
програми

Трифон Трифонов

Структури от данни и програмиране,
спец. Компютърни науки, 2 поток, 2015/16 г.

16 октомври 2015 г.

Типове данни (ТД)

Инструмент за класификация на данните, характеризиращ се с:

- множество от стойности
- операции над стойностите

Типове данни (ТД)

Инструмент за класификация на данните, характеризиращ се с:

- множество от стойности
- операции над стойностите

За какво служат типовете?

Типове данни в C++

Примитивни:

- булев (`bool`)
- целочислен (`short`, `int`, `long`, `unsigned`)
- числа с плаваща запетая (`float`, `double`)
- символен (`char`)
- указател (`*`)
- псевдоним (`&`)

Съставни:

- масив (`[]`)
- структура / запис (`struct`)
- клас (`class`)

Структури от данни (СД)

- СД са схеми за организация на даден вид данни в паметта на компютъра

Структури от данни (СД)

- СД са схеми за организация на даден вид данни в паметта на компютъра
  - обикновено с цел ефективност

Структури от данни (СД)

- СД са схеми за организация на даден вид данни в паметта на компютъра
  - обикновено с цел ефективност
- Всеки съставен ТД в частност може да се разглежда като СД

Структури от данни (СД)

- СД са схеми за организация на даден вид данни в паметта на компютъра
  - обикновено с цел ефективност
- Всеки съставен ТД в частност може да се разглежда като СД
  - **Пример:** `int[]` е ТД масив от елементи от тип `int` и може да се разглежда като СД, която представя редица от цели числа последователно в паметта

Структури от данни (СД)

- СД са схеми за организация на даден вид данни в паметта на компютъра
  - обикновено с цел ефективност
- Всеки съставен ТД в частност може да се разглежда като СД
  - **Пример:** `int[]` е ТД масив от елементи от тип `int` и може да се разглежда като СД, която представя редица от цели числа последователно в паметта
- СД често се реализират чрез потребителски дефинирани ТД

Структури от данни (СД)

- СД са схеми за организация на даден вид данни в паметта на компютъра
  - обикновено с цел ефективност
- Всеки съставен ТД в частност може да се разглежда като СД
  - **Пример:** `int[]` е ТД масив от елементи от тип `int` и може да се разглежда като СД, която представя редица от цели числа последователно в паметта
- СД често се реализират чрез потребителски дефинирани ТД
  - **Пример:** СД "разширяващ се стек" може да се реализира с ТД клас `ResizingStack`

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите
- Не налага конкретна организация на паметта

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите
- Не налага конкретна организация на паметта
  - (за разлика от СД)

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите
- Не налага конкретна организация на паметта
  - (за разлика от СД)
- Не налага конкретно представяне със средствата на някакъв език

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите
- Не налага конкретна организация на паметта
  - (за разлика от СД)
- Не налага конкретно представяне със средствата на някакъв език
  - (за разлика от ТД)

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите
- Не налага конкретна организация на паметта
  - (за разлика от СД)
- Не налага конкретно представяне със средствата на някакъв език
  - (за разлика от ТД)
- Допуска една или повече реализации

Абстрактен тип данни (АТД)

- Формален модел на ТД или СД
  - Множество от стойности
  - Описание на имената и вида на операциите
  - Описание на поведението и свойствата на операциите
- Не налага конкретна организация на паметта
  - (за разлика от СД)
- Не налага конкретно представяне със средствата на някакъв език
  - (за разлика от ТД)
- Допуска една или повече реализации
- Какви са предимствата на АТД?

Видове описания на СД

Логическо описание (АТД)

- същност и предназначение
- компоненти
- операции
- свойства на операциите

Физическо описание

- организация на паметта
- представяне с един или повече ТД
- реализация на операциите

Видове СД

- Според вида на комопонентите си
  - хомогенни (масив)
  - хетерогенни (структура)
- Според способността за промяна на размера
  - статични (статичен масив
  - динамични (разширяващ се масив)
- Според вътрешната структура
  - линейни (свързан списък)
  - разклонени (дърво)
  - мрежови (граф)

Какво е алгоритъм?

Какво е алгоритъм?

Неформално:
Добре дефиниран набор от инструкции за извършване на дадено пресмятане.

Какво е алгоритъм?

Неформално:
Добре дефиниран набор от инструкции за извършване на дадено пресмятане.

Формално:
Машина на Тюринг

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)
- Алгоритъмът като решение на масова задача

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)
- Алгоритъмът като решение на масова задача
- Една масова задача може да има много възможни решения

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)
- Алгоритъмът като решение на масова задача
- Една масова задача може да има много възможни решения
- Как да сравняваме алгоритмите?

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)
- Алгоритъмът като решение на масова задача
- Една масова задача може да има много възможни решения
- Как да сравняваме алгоритмите?
- Добре е да имаме мярка за ефективността на алгоритъма

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)
- Алгоритъмът като решение на масова задача
- Една масова задача може да има много възможни решения
- Как да сравняваме алгоритмите?
- Добре е да имаме мярка за ефективността на алгоритъма

Масови задачи

- Масова задача: общ изчислителен проблем, който може да бъде формулиран за входни данни с произволен размер
  - Пример: подреждане на елементи на масив във възходящ ред (сортиране)
- Алгоритъмът като решение на масова задача
- Една масова задача може да има много възможни решения
- Как да сравняваме алгоритмите?
- Добре е да имаме мярка за ефективността на алгоритъма

Видове сложност

- времева сложност — оценка на времето за изпълнение на алгоритъма
- пространствена сложност — оценка на паметта използвана от алгоритъма

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!
- брой “атомарни операции” и брой “единици памет”?

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!
- брой “атомарни операции” и брой “единици памет”?
  - зависи колко големи данни подадем на алгоритъма

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!
- брой “атомарни операции” и брой “единици памет”?
  - зависи колко големи данни подадем на алгоритъма
- функция на броя операции или променливи в зависимост от големината на входа?

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!
- брой “атомарни операции” и брой “единици памет”?
  - зависи колко големи данни подадем на алгоритъма
- функция на броя операции или променливи в зависимост от големината на входа?
  - точният брой може да варира в зависимост от конкретната реализация или език за програмиране

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!
- брой "атомарни операции" и брой "единици памет"?
  - зависи колко големи данни подадем на алгоритъма
- функция на броя операции или променливи в зависимост от големината на входа?
  - точният брой може да варира в зависимост от конкретната реализация или език за програмиране
  - но варирането **не трябва да зависи от големината на входа**

Сложност на алгоритъм

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?
  - не знаем на какъв процесор ще се изпълнява програмата!
- брой "атомарни операции" и брой "единици памет"?
  - зависи колко големи данни подадем на алгоритъма
- функция на броя операции или променливи в зависимост от големината на входа?
  - точният брой може да варира в зависимост от конкретната реализация или език за програмиране
  - но варирането **не трябва да зависи от големината на входа**

Асимптотична сложност

Оценка на функцията на сложност при произволно голям вход

O-нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности

Решение: дефинираме класове от функции

O-нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности
Решение: дефинираме класове от функции

Нека $f, g : \mathbb{N} \to \mathbb{N}$.

$$f \in O(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n)) \quad (f \text{ расте по-бавно от } g)$$

O-нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности
Решение: дефинираме класове от функции

Нека $f, g : \mathbb{N} \to \mathbb{N}$.

$$f \in O(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n)) \quad (f \text{ расте по-бавно от } g)$$
$$f \in \Omega(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n)) \quad (f \text{ расте по-бързо от } g)$$

O-нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности
Решение: дефинираме класове от функции

Нека $f, g : \mathbb{N} \to \mathbb{N}$.

$$f \in O(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n)) \quad (f \text{ расте по-бавно от } g)$$
$$f \in \Omega(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n)) \quad (f \text{ расте по-бързо от } g)$$

Лесно се вижда, че $f \in O(g) \Leftrightarrow g \in \Omega(f)$.

O-нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности
Решение: дефинираме класове от функции

Нека $f, g : \mathbb{N} \to \mathbb{N}$.

$$f \in O(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n)) \quad (f \text{ расте по-бавно от } g)$$
$$f \in \Omega(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n)) \quad (f \text{ расте по-бързо от } g)$$

Лесно се вижда, че $f \in O(g) \Leftrightarrow g \in \Omega(f)$.

$$f \in \theta(g) \Leftrightarrow \exists C_1 \exists C_2 \exists k \forall n \geq k (0 \leq C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n))$$

O-нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности
Решение: дефинираме класове от функции

Нека $f, g : \mathbb{N} \to \mathbb{N}$.

$$f \in O(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n)) \quad (f \text{ расте по-бавно от } g)$$
$$f \in \Omega(g) \Leftrightarrow \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n)) \quad (f \text{ расте по-бързо от } g)$$

Лесно се вижда, че $f \in O(g) \Leftrightarrow g \in \Omega(f)$.

$$f \in \theta(g) \Leftrightarrow \exists C_1 \exists C_2 \exists k \forall n \geq k (0 \leq C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n))$$

Лесно се вижда, че $\Theta(g) = O(g) \cap \Omega(g)$.

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)
- $3n^2 + 100 \in \Theta(n^2)$

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)
- $3n^2 + 100 \in \Theta(n^2)$, защото при $C_1 = 1, C_2 = 4, k = 10$ е вярно, че

$$\forall n \geq 10 (0 \leq n^2 \leq 3n^2 + 100 \leq 4n^2)$$

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)
- $3n^2 + 100 \in \Theta(n^2)$, защото при $C_1 = 1, C_2 = 4, k = 10$ е вярно, че

$$\forall n \geq 10 (0 \leq n^2 \leq 3n^2 + 100 \leq 4n^2)$$

- $2^n \in O(3^n)$, $\log_2(n) \in \Theta(log_{10}(n))$

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)
- $3n^2 + 100 \in \Theta(n^2)$, защото при $C_1 = 1, C_2 = 4, k = 10$ е вярно, че

$$\forall n \geq 10 (0 \leq n^2 \leq 3n^2 + 100 \leq 4n^2)$$

- $2^n \in O(3^n)$, $\log_2(n) \in \Theta(log_{10}(n))$
  - обикновено бележим просто $\log n$, понеже основата няма значение

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)
- $3n^2 + 100 \in \Theta(n^2)$, защото при $C_1 = 1, C_2 = 4, k = 10$ е вярно, че

$$\forall n \geq 10 (0 \leq n^2 \leq 3n^2 + 100 \leq 4n^2)$$

- $2^n \in O(3^n)$, $\log_2(n) \in \Theta(log_{10}(n))$
  - обикновено бележим просто $\log n$, понеже основата няма значение
- $\log n \in O(n^{0.001})$, $n^{1000} \in O(1.001^n)$

O-нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (при $C = 1000$ е вярно, че $1000n^2 \leq 1000n^3$)
- $3n^2 + 100 \in \Theta(n^2)$, защото при $C_1 = 1, C_2 = 4, k = 10$ е вярно, че

$$\forall n \geq 10(0 \leq n^2 \leq 3n^2 + 100 \leq 4n^2)$$

- $2^n \in O(3^n)$, $\log_2(n) \in \Theta(log_{10}(n))$
  - обикновено бележим просто $\log n$, понеже основата няма значение
- $\log n \in O(n^{0.001})$, $n^{1000} \in O(1.001^n)$
- $\forall C(C \in \Theta(1))$, $O(1) = \Theta(1)$, $\Omega(1) = \mathbb{N}^{\mathbb{N}}$

Видове сложност

- В най-лошия случай (песимистична)

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?
- В най-добрия случай (оптимистична)

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?
- В най-добрия случай (оптимистична)
  - Какъв е минималният възможен брой операции (единици памет), които може да извърши (използва) алгоритъмът, за да реши задачата?

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?
- В най-добрия случай (оптимистична)
  - Какъв е минималният възможен брой операции (единици памет), които може да извърши (използва) алгоритъмът, за да реши задачата?
- В средния случай (средна)

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?
- В най-добрия случай (оптимистична)
  - Какъв е минималният възможен брой операции (единици памет), които може да извърши (използва) алгоритъмът, за да реши задачата?
- В средния случай (средна)
  - Ако считаме, че всеки възможен вход е равновероятен, какво е "средното аритметично" на броя операции (единици памет), които трябват при всички възможни входове?

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?
- В най-добрия случай (оптимистична)
  - Какъв е минималният възможен брой операции (единици памет), които може да извърши (използва) алгоритъмът, за да реши задачата?
- В средния случай (средна)
  - Ако считаме, че всеки възможен вход е равновероятен, какво е "средното аритметично" на броя операции (единици памет), които трябват при всички възможни входове?
- В контекста на конкретна програма (амортизирана)

Видове сложност

- В най-лошия случай (песимистична)
  - Какъв е максималният възможен брой операции (единици памет), които могат да са нужни на алгоритъма, за да реши задачата?
- В най-добрия случай (оптимистична)
  - Какъв е минималният възможен брой операции (единици памет), които може да извърши (използва) алгоритъмът, за да реши задачата?
- В средния случай (средна)
  - Ако считаме, че всеки възможен вход е равновероятен, какво е "средното аритметично" на броя операции (единици памет), които трябват при всички възможни входове?
- В контекста на конкретна програма (амортизирана)
  - Ако алгоритъмът ще се извиква няколко пъти в рамките на дадена програма, колко операции (единици памет) средно ще са му необходими за едно извикване?

Пресмятане на сложност: пример

```
for(int i = 0; i < n-1; i++)
  for(int j = n-2; j >= i; j--)
    if (a[j] > a[j+1]) {
      double x = a[j];
      a[j] = a[j+1];
      a[j+1]= x;
    }
```

Да се оценят песимистична, оптимистична и средна времева сложност.

Standard Template Library (STL)

Библиотека от шаблони, реализираща стандартни структури от данни и алгоритми.

- част от C++ Standard Library
- основни компоненти
  - алгоритми (`<algorithm>`)
  - контейнери (`<queue>`, `<stack>`, `<vector>`, `<list>`, `<map>`, ...
  - функционални обекти (`<functional>`)
- базирана на идеята за генерично програмиране
- ефективност: дава гаранции за сложност на алгоритми и операции над СД