

Структури от данни
+
алгоритми
=
програми

Трифон Трифонов

Структури от данни и програмиране, спец. Компютърни науки, 2 поток, 2024/25 г.

3 октомври 2024 г.

Тази презентация е достъпна под лиценза Creative Commons Признание-Некомерсиално-Споделяне на споделеното 4.0 Международен





Niklaus Wirth.

Algorithms + data structures = programs.

Englewood Cliffs, N.J. : Prentice-Hall, 1976.

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

Person p = "Иван";

За какво служат типовете?

char const* s = Person(...);

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 `class ResizingStack`

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
 - (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
 - (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 описание на СД

Логическо описание (АТД)

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

Видове описание 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 компонентите си
 - хомогенни (масив)
 - хетерогенни (структура)

Видове СД

- Според вида на компонентите си
 - хомогенни (масив)
 - хетерогенни (структура)
- Според способността за промяна на размера
 - статични (статичен масив)
 - динамични (разширяващ се масив)

Видове СД

- Според вида на компонентите си
 - хомогенни (масив)
 - хетерогенни (структура)
- Според способността за промяна на размера
 - статични (статичен масив)
 - динамични (разширяващ се масив)
- Според връзките между данните
 - 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 (например)

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
- Такава мярка обикновено се нарича **сложност на алгоритъма**

Сложност на алгоритъм

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Сложност на алгоритъм

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

- **времева** — оценка на времето за изпълнение на алгоритъма

Сложност на алгоритъм

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

Сложност на алгоритъм

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

Как да оценим колко ресурс използва даден алгоритъм?

Сложност на алгоритъм

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

Как да оценим колко ресурс използва даден алгоритъм?

- брой процесорни инструкции и брой байтове?

Сложност на алгоритъм

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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

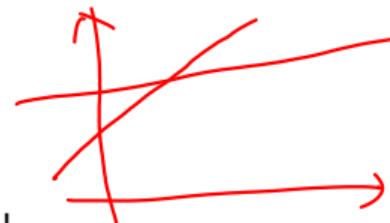
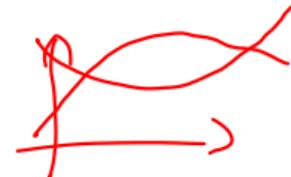
Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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

Алгоритъм е по-ефективен, ако има нужда от по-малко ресурс.

Видове сложност:

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

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 нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз
- $f_A(10) = 130$, $f_B(10) = 1005$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз
- $f_A(10) = 130$, $f_B(10) = 1005$
- $f_A(100) = 10300$, $f_B(100) = 10005$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз
- $f_A(10) = 130, \quad f_B(10) = 1005$
- $f_A(100) = 10300, \quad f_B(100) = 10005$
- $f_A(1000) = 1003000, \quad f_B(1000) = 100005$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при **нарастването** на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз
- $f_A(10) = 130, \quad f_B(10) = 1005$
- $f_A(100) = 10300, \quad f_B(100) = 10005$
- $f_A(1000) = 1003000, \quad f_B(1000) = 100005$
- Но нас ни интересува какво се случва при **големи данни ($n \rightarrow \infty$)!**

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз
- $f_A(10) = 130, \quad f_B(10) = 1005$
- $f_A(100) = 10300, \quad f_B(100) = 10005$
- $f_A(1000) = 1003000, \quad f_B(1000) = 100005$
- Но нас ни интересува какво се случва при **големи данни ($n \rightarrow \infty$)!**
- За големи данни B е все по-бърз в сравнение с A

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 1:

- Нека имаме два алгоритъма A и B .
- При вход с големина n двата алгоритъма изпълняват съответно $f_A(n) = n^2 + 3n$ операции и $f_B(n) = 100n + 5$ операции.
- Кой от двата алгоритъма е по-бърз
- $f_A(10) = 130, f_B(10) = 1005$
- $f_A(100) = 10300, f_B(100) = 10005$
- $f_A(1000) = 1003000, f_B(1000) = 100005$
- Но нас ни интересува какво се случва при **големи данни ($n \rightarrow \infty$)!**
- За големи данни B е все по-бърз в сравнение с A
- За големи данни $f_A(n) \approx n^2, f_B(n) \approx 100n$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$
- Изобщо, за $n \geq 10$ имаме $f_C(n) \leq f_D(n)$

Асимптотична сложност

Оценка на **нарастването** на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$
- Изобщо, за $n \geq 10$ имаме $f_C(n) \leq f_D(n)$
- Но нас ни интересува **нарастването** на функцията!

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$
- Изобщо, за $n \geq 10$ имаме $f_C(n) \leq f_D(n)$
- Но нас ни интересува **нарастването** на функцията!
- $f_A(2n) = 4n^2 + 6n \approx 4f_A(n)$, $f_B(2n) = 200n + 5 \approx 2f_B(n)$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$
- Изобщо, за $n \geq 10$ имаме $f_C(n) \leq f_D(n)$
- Но нас ни интересува **нарастването** на функцията!
- $f_A(2n) = 4n^2 + 6n \approx 4f_A(n)$, $f_B(2n) = 200n + 5 \approx 2f_B(n)$
- $f_C(2n) \approx 4f_C(n)$, $f_D(2n) \approx 4f_D(n)$

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$
- Изобщо, за $n \geq 10$ имаме $f_C(n) \leq f_D(n)$
- Но нас ни интересува **нарастването** на функцията!
- $f_A(2n) = 4n^2 + 6n \approx 4f_A(n)$, $f_B(2n) = 200n + 5 \approx 2f_B(n)$
- $f_C(2n) \approx 4f_C(n)$, $f_D(2n) \approx 4f_D(n)$
- A, C, D се усложняват с еднаква скорост с увеличаване на n , а B се усложнява по-бавно от тях.

Асимптотична сложност

Оценка на нарастването на функцията на сложност при нарастването на големината на данните за обработка.

Пример 2:

- Нека C и D имат сложности $f_C(n) = 2n^2 + 10n$, $f_D(n) = 3n^2$.
- $f_C(5) = 100$, $f_D(5) = 75$
- $f_C(10) = f_D(10) = 300$
- $f_C(20) = 1000$, $f_D(20) = 1200$
- Изобщо, за $n \geq 10$ имаме $f_C(n) \leq f_D(n)$
- Но нас ни интересува **нарастването** на функцията!
- $f_A(2n) = 4n^2 + 6n \approx 4f_A(n)$, $f_B(2n) = 200n + 5 \approx 2f_B(n)$
- $f_C(2n) \approx 4f_C(n)$, $f_D(2n) \approx 4f_D(n)$
- A, C, D се усложняват с еднаква скорост с увеличаване на n , а B се усложнява по-бавно от тях.
- Искаме класификация, която нарежда f_A , f_C и f_D на едно и също ниво, а f_B по-ниско.

O -нотация

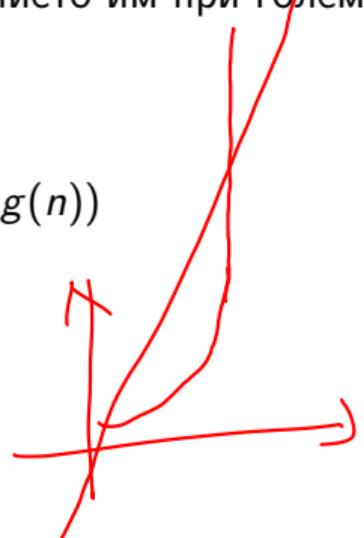
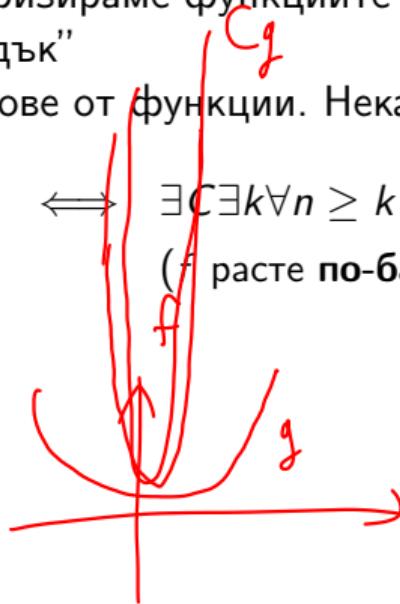
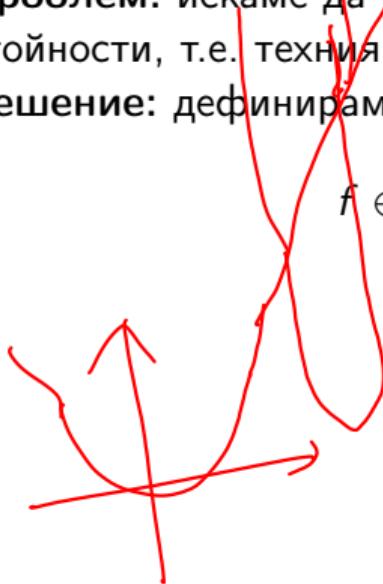
Проблем: искаме да категоризираме функциите относно поведението им при големи стойности, т.е. техния “порядък”

Решение: дефинираме класове от функции.

O -нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности, т.е. техния "порядък"

Решение: дефинираме класове от функции. Нека $f, g : \mathbb{N} \rightarrow \mathbb{N}$.



$$f \in O(g) \iff \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n))$$

(f расте **по-бавно** от g)

O -нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности, т.е. техния “порядък”

Решение: дефинираме класове от функции. Нека $f, g : \mathbb{N} \rightarrow \mathbb{N}$.

$$f \in O(g) \iff \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n))$$

(f расте **по-бавно** от g)

$$f \in \Omega(g) \iff \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n))$$

(f расте **по-бързо** от g)

O -нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности, т.е. техния “порядък”

Решение: дефинираме класове от функции. Нека $f, g : \mathbb{N} \rightarrow \mathbb{N}$.

$$f \in O(g) \iff \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n))$$

(f расте **по-бавно** от g)

$$f \in \Omega(g) \iff \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n))$$

(f 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, т.е. техния “порядък”

Решение: дефинираме класове от функции. Нека $f, g : \mathbb{N} \rightarrow \mathbb{N}$.

$$f \in O(g) \iff \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n))$$

*(f расте **по-бавно** от g)*

$$f \in \Omega(g) \iff \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n))$$

*(f расте **по-бързо** от g)*

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

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

O -нотация

Проблем: искаме да категоризираме функциите относно поведението им при големи стойности, т.е. техния “порядък”

Решение: дефинираме класове от функции. Нека $f, g : \mathbb{N} \rightarrow \mathbb{N}$.

$$f \in O(g) \iff \exists C \exists k \forall n \geq k (0 \leq f(n) \leq C \cdot g(n))$$

*(f расте **по-бавно** от g)*

$$f \in \Omega(g) \iff \exists C \exists k \forall n \geq k (0 \leq C \cdot g(n) \leq f(n))$$

*(f расте **по-бързо** от g)*

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

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

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

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

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

$$\log_a n = \frac{\log_b n}{\log_b a}$$

$\log_b a$

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
- $\log n \in O(n^{0.001}), n^{1000} \in O(1.001^n)$

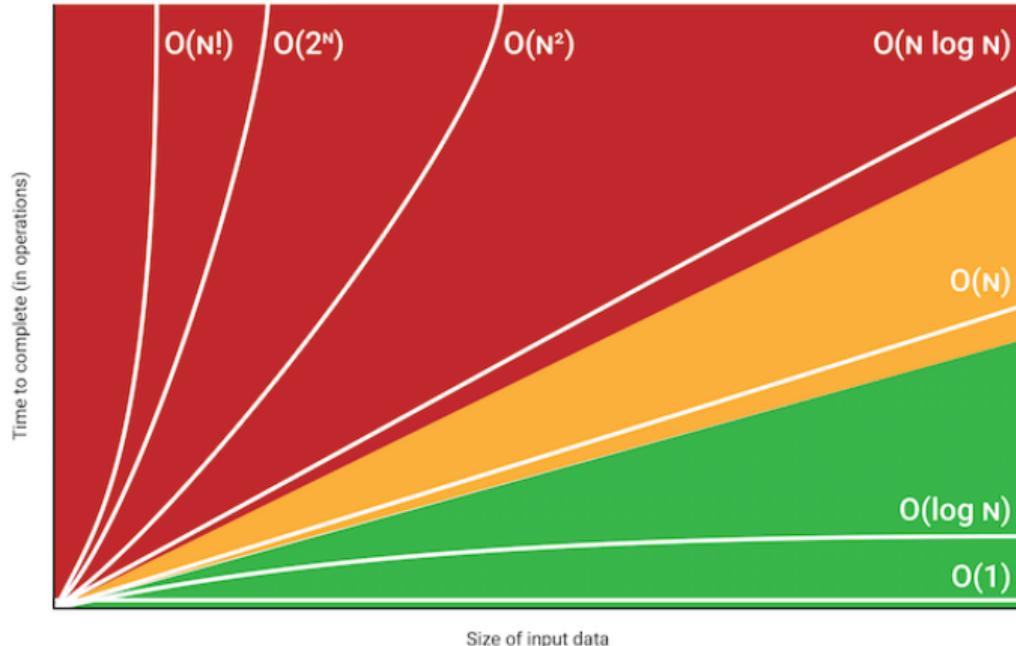
O -нотация: примери

- $f \in \Theta(f) = O(f) \cap \Omega(f)$
- $1000n^2 \in O(n^3)$ (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
- $\log n \in O(n^{0.001}), n^{1000} \in O(1.001^n)$
- $\forall C (C \in \Theta(1)), O(1) = \Theta(1), \Omega(1) = \mathbb{N}^\mathbb{N}$

O -нотация: често използвани сложности



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-добраия 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-добраия 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?

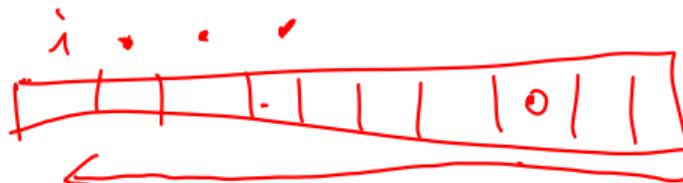
Видове сложност

- 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?
- При многократно изпълнение (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-добраия 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?
- При многократно изпълнение (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;
        }
    }
}

```

$$\begin{aligned}
 f(i, n) &= 3(n - 1 - i) + 1 + 1 \\
 &= 6n - 36i - 4 \\
 g(n) &= f(0, n) + f(1, n) + \dots + f(n-1, n) \\
 &\quad + 2n + 1
 \end{aligned}$$

Да се оценят пессимистична, оптимистична и средна времева сложност.

$$= \sum_{i=0}^{n-1} f(i, n) + 2n + 1$$

Пресмятане на сложност: пример

$$\begin{aligned}
 &= \sum_{i=0}^{n-1} f(i, n) + 2n + 1 = \\
 \text{for}(int i = 0; i < n-1; i++) \\
 \text{ for}(int j = n-2; j \geq i; j--) \\
 \text{ if } (a[j] > a[j+1]) \{ \\
 \text{ double x = a[j];} \\
 \text{ a[j] = a[j+1];} \\
 \text{ a[j+1] = x;} \\
 \text{ }\} \\
 &= \sum_{i=0}^{n-1} [6n - 4i - 4] + 2n + 1 \\
 &= n(6n - 4) - 4 \sum_{i=0}^{n-1} i + 2n + 1
 \end{aligned}$$

Да се оценят пессимистична, оптимистична и средна времева сложност.

$$\overset{\nearrow}{O(n^2)}$$

$$\overset{\nearrow}{\Theta(n^2)} \quad \frac{n(n-1)}{2}$$

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