Teoretyczne podstawy informatyki

Wykład 6b:
Rozwiązywanie rekurencji

http://hibiscus.if.uj.edu.pl/~erichter/Dydaktyka2009/TPI-2009

Czas działania programu

- Dla konkretnych danych wejściowych jest wyrażony liczba wykonanych prostych (elementarnych) operacji lub "kroków". Jest dogodne zrobienie założenia że operacja elementarna jest maszynowo niezależna.
- Każde wykonanie **i**-tego wiersza programu jest równe c_i, przy czym c_i jest stałą.
- Kiedy algorytm zawiera **rekurencyjne wywołanie samego siebie**, jego czas działania można często opisać zależnością rekurencyjna (rekurencja) wyrażającą czas dla problemu rozmiaru **n** za pomocą czasu dla podproblemów mniejszych rozmiarów.
- Możemy wiec użyć narzędzi matematycznych aby rozwiązać rekurencje i w ten sposób otrzymać oszacowania czasu działania algorytmu.

Rekurencja dla algorytmu typu "dziel i zwyciezaj"

- Rekurencja odpowiadającą czasowi działania algorytmu typu "dziel i zwyciężaj" opiera się na podziale jednego poziomu rekursji na trzy etapy.
  - Niech **T(n)** będzie czasem działania dla jednego problemu rozmiaru **n**.
  - Jeśli rozmiar problemu jest odpowiednio mały, powiedzmy **n ≤ c** dla pewnej stałej **c**, to jego rozwiązanie zajmuje stały czas, co zapiszemy jako **Θ(1)**.
  - Załóżmy ze dzielimy problem na **a** podproblemów, każdy rozmiaru **n/b**. Jeśli **D(n)** jest czasem dzielenia problemu na podproblemy, a **C(n)** czasem scalania rozwiązań podproblemow w pełne rozwiązanie dla oryginalnego problemu, to otrzymujemy rekurencje
  - **T(n) = Θ(1)** **jeśli n ≤ c**
  - **T(n) = a T(n/b) + D(n) + C(n)** **w przeciwnym przypadku**

Rekurencja dla algorytmu typu "dziel i zwyciężaj"

- **Przykład:** algorytm sortowania przez scalanie
  - **dziel:** znajdujemy środek przedziału, zajmuje to czas stały **$D(n)=\Theta(1)$,**
  - **zwyciężaj:** rozwiązujemy rekurencyjnie dwa podproblemy, każdy rozmiaru **$n/2$**, co daje czas działania **$2\,T(n/2)$**,
  - **połącz:** działa w czasie **$\Theta(n)$**, a wiec **$C(n)=\Theta(n)$**.
- **Ostatecznie:**
  - **$T(n) = \Theta(1)$** **jeśli $n=1$**
  - **$T(n) = 2\,T(n/2) + \Theta(1) + \Theta(n)$** **jeśli $n>1$**
- Rozwiązaniem tej rekurencji jest **$T(n) = \Theta(n \log n)$**.

Metody rozwiązywania rekurencji

- **Metoda podstawiania:**
  - zgadujemy oszacowanie, a następnie dowodzimy przez indukcję jego poprawność.
- **Metoda iteracyjna:**
  - przekształcamy rekurencję na sumę, korzystamy z technik ograniczania sum.
- **Metoda uniwersalna:**:
  - stosujemy oszacowanie na rekurencję mające postać **$T(n) = a\,T(n/b) + f(n)$**, gdzie **$a \geq 1$**, **$b > 1$**, a **$f(n)$** jest daną funkcją.

Metoda podstawiania

- Polega na zgadnięciu postaci rozwiązania, a następnie wykazaniu przez indukcję, że jest ono poprawne. Trzeba też znaleźć odpowiednie stałe. Bardzo skuteczna, stosowana tylko w przypadkach kiedy łatwo jest przewidzieć postać rozwiązania.

Metoda podstawiania

- **Przykład:**
  - Postać rekurencji:
    $T(n) = 2T(n/2) + n$
  - Zgadnięte rozwiązanie:
    $T(n) = \Theta(n \log n)$
  - Podstawa:
    $n=2; T(1)=1; T(2)=4;$
  - Indukcja:
    $T(n) \leq 2\,(c(n/2)\log(n/2)) + n \leq c\,n \log(n/2) + n$
    $T(n) \leq c\,n \log(n/2) + n = cn \log(n) - cn \log(2) + n$
    $T(n) \leq cn \log(n) - cn \log(2) + n = cn \log(n) - cn + n$
    $T(n) \leq cn \log(n) - cn + n \leq cn \log(n)$

  spełnione dla $c>=1;$

Metoda iteracyjna

- Polega na rozwijaniu (iterowaniu) rekurencji i wyrażanie jej jako sumy składników zależnych tylko od **n** warunków brzegowych. Następnie mogą być użyte techniki sumowania do oszacowania rozwiązania.

Metoda iteracyjna

- **Przykład:**
  - Postać rekurencji:
    $T(n) = 3T(n/4) + n$
  - Iterujemy:
    $T(n) = n + 3T(n/4) = n + 3((n/4) + 3T(n/16)) = n + 3\,(n/4) + 9T(n/16)$
    $T(n) = n + 3\,(n/4) + 9T(n/16) = n + 3\,n/4 + 9\,n/16 + 27\,T(n/64)$
- Iterujemy tak długo aż osiągniemy warunki brzegowe.
  Składnik **i**-ty w ciągu wynosi $\mathbf{3^i\ n/4^i}$.
  Iterowanie kończymy, gdy $\mathbf{n=1}$ lub $\mathbf{n/4^i = 1}$ (czyli $\mathbf{i > \log_4(n)}$).
  $$\mathbf{T(n) \le n + 3n/4 + 9n/16 + 27n/64 + \ldots + 3^{\log_4 n}\ \Theta(1)}$$
  $$\mathbf{T(n) \le 4n + 3^{\log_4 n}\ \Theta(1) = \Theta(n)}$$

Metoda iteracyjna

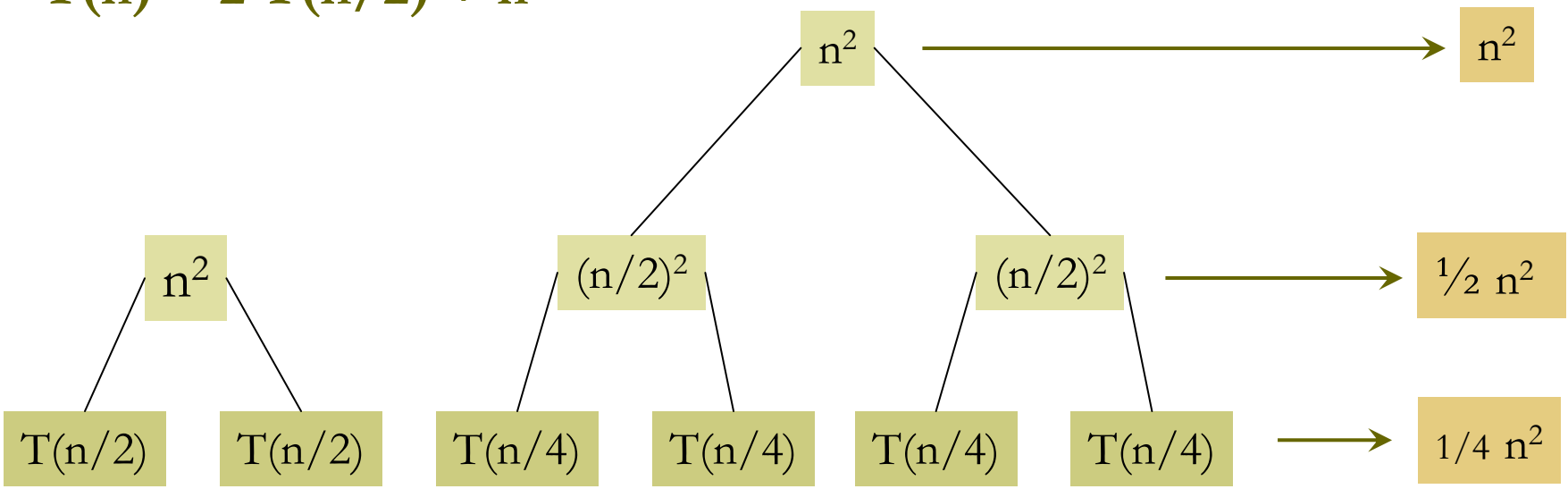
- Metoda iteracyjna jest zazwyczaj związana z dużą ilością przekształceń algebraicznych, więc zachowanie prostoty nie jest łatwe.
- Punkt kluczowy to skoncentrowanie się na dwóch parametrach:
  - liczbie iteracji koniecznych do osiągnięcia warunku brzegowego
  - oraz sumie składników pojawiających się w każdej iteracji.

Drzewa rekursji

- Pozwalają w dogodny sposób zilustrować rozwijanie rekurencji, jak również ułatwia stosowanie aparatu algebraicznego służącego do rozwiązywania tej rekurencji.
- Szczególnie użyteczne gdy rekurencja opisuje algorytm typu “dziel i zwyciężaj”.

Drzewo rekursji dla algorytmu „dziel i zwyciężaj

$$T(n) = 2\,T(n/2) + n^2$$

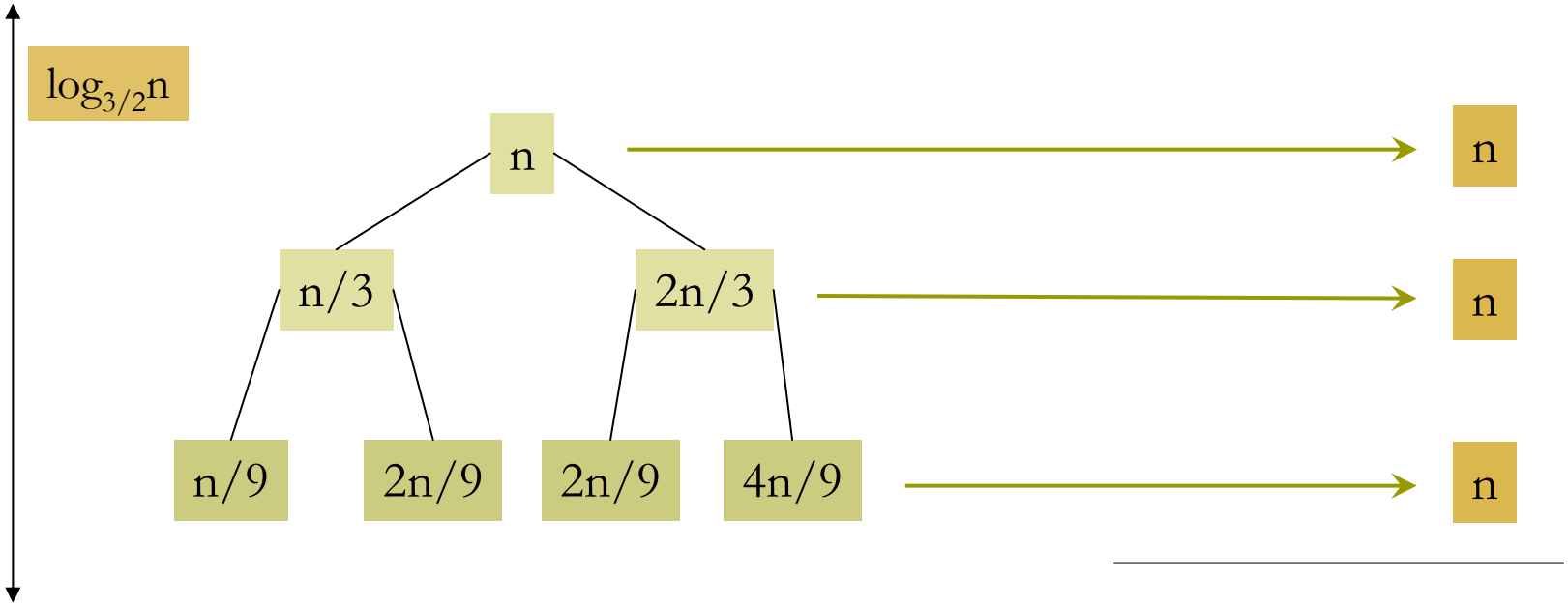


ostateczny wynik: $T(n) = \Theta(n^2)$

w sumie: $\Theta(n^2)$

Drzewa rekursji

$$T(n) = T(n/3) + T(2n/3) + n$$



wysokość drzewa: $\log_{3/2}(n) < \log(n)$

w sumie $\Theta(n \log(n))$

ostateczny wynik: $T(n) = \Theta(n \log(n))$

Metoda rekurencji uniwersalnej

- Metoda rekurencji uniwersalnej podaje "uniwersalny przepis" rozwiązywania równania rekurencyjnego postaci:
  - **$T(n) = a\,T(n/b) + f(n)$**
- gdzie **$a \geq 1$** i **$b > 1$** są stałymi, a **$f(n)$** jest funkcja asymptotycznie dodatnia.
- Za wartość **(n/b)** przyjmujemy najbliższą liczbę całkowitą (mniejsza lub większą od wartości dokładnej).

Metoda rekurencji uniwersalnej

- Rekurencja opisuje czas działania algorytmu, który dzieli problem rozmiaru **n** na **a** problemów, każdy rozmiaru **n/b**, gdzie **a** i **b** są dodatnimi stałymi.
- Każdy z **a** problemów jest rozwiązywany rekurencyjnie w czasie **T(n/b)**.
- Koszt dzielenia problemu oraz łączenia rezultatów częściowych jest opisany funkcja **f(n)**.

Twierdzenie o rekurencji uniwersalnej

- Niech **$a \geq 1$** i **$b > 1$** będą stałymi, niech **$f(n)$** będzie pewną funkcją i niech **$T(n)$** będzie zdefiniowane dla nieujemnych liczb całkowitych przez rekurencje

  $$T(n) = a\,T(n/b) + f(n)$$

  gdzie **(n/b)** oznacza najbliższą liczbę całkowitą do wartości dokładnej **n/b**.
- Wtedy funkcja **$T(n)$** może być ograniczona asymptotycznie w następujący sposób:
  - Jeśli **$f(n) = O(n^{\log_b a - \varepsilon})$** dla pewnej stałej **$\varepsilon > 0$**, to **$T(n) = \Theta(n^{\log_b a})$**.
  - Jeśli **$f(n) = \Theta(n^{\log_b a})$** to **$T(n) = \Theta(n^{\log_b a} \log n)$**.
  - Jeśli **$f(n) = n^{\log_b a + \varepsilon})$** dla pewnej stałej **$\varepsilon > 0$** i jeśli **$af(n/b) \leq cf(n)$** dla pewnej stałej **$c < 1$** i wszystkich dostatecznie dużych **n**, to **$T(n) = Q(f(n))$**.

Twierdzenie o rekurencji uniwersalnej

- **"Intuicyjnie…":**
  - W każdym z trzech przypadków porównujemy funkcje **f(n)** z funkcją $\mathbf{n^{\log_b a}}$. Rozwiązanie rekurencji zależy od większej z dwóch funkcji.
  - Jeśli funkcja $\mathbf{n^{\log_b a}}$ jest większa, to rozwiązaniem rekurencji jest:
    - $\mathbf{T(n) = \Theta(n^{\log_b a})}$.
  - Jeśli **f(n)** jest większa, to rozwiązaniem jest:
    - $\mathbf{T(n) = \Theta(f(n))}$.
  - Jeśli funkcje są tego samego rzędu, to mnożymy przez **log n** i rozwiązaniem jest:
    - $\mathbf{T(n) = \Theta(n^{\log_b a} \log n) = T(n) = \Theta(f(n) \log n)}$.

Przykład

- **$T(n) = 9\,T(n/3) + n$**

  $a=9$,
  $b=3$,
  $f(n)=n$,

  a zatem **$n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$**.

- Ponieważ **$f(n)=O(n^{\log_3 9-\varepsilon})$**, gdzie **$\varepsilon=1$**, możemy zastosować przypadek 1 z twierdzeniai wnioskować że rozwiązaniem jest **$T(n) = \Theta(n^2)$**.

Przykład

- $T(n) = T(2n/3) + 1$

  $a=1,$
  $b=3/2,$
  $f(n)=1,$

  a zatem $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$.

- Stosujemy przypadek 2, gdyż $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, a zatem rozwiązaniem rekurencji jest $T(n) = \Theta(\log n)$.

Przykład

- **$T(n) = 3T(n/4) + n \log n$**

  $a=3$,
  $b=4$,
  $f(n)=n \log n$,

  a zatem **$n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$**.
- Ponieważ **$f(n) = \Omega(n^{\log_4 3+\varepsilon})$** , gdzie **$\varepsilon \approx 0.2$**, wiec stosuje się tutaj przypadek 3, jeśli możemy pokazać ze dla **f(n)** zachodzi warunek regularności.
  Dla dostatecznie dużych **n**:

  $af(n/b) = 3(n/4)\log(n/4) \leq (3/4)n\log(n) = c\, f(n)$

  dla **$c=3/4$**.
- Warunek jest spełniony i możemy napisać że rozwiązaniem rekurencji jest **$T(n) = \Theta(n \log n)$**.

Przykład

- **$T(n) = 2T(n/2) + n \log n$**

  $a=2,$
  $b=2,$
  $f(n)=n \log n,$

  a zatem **$n^{\log_b a} = n$**.
- Wydaje się że powinien to być przypadek 3, gdyż **$f(n)=n \log n$** jest asymptotycznie większe niż **$n^{\log_b a} = n$**, ale nie wielomianowo większy.
- Stosunek **$f(n)/ n^{\log_b a} = (n \log n)/n = \log n$** jest asymptotycznie mniejszy niż **n^{ε}** dla każdej dodatniej stałej **ε**.
- W konsekwencji rekurencja ta "wpada" w lukę miedzy przypadkiem 2 i 3.

Rekurencja

- Rekurencje były badane już w 1202 roku przez L. Fibonacciego, od którego nazwiska pochodzi nazwa liczb Fibonacciego.
- A. De Moivre w 1730 roku wprowadził pojęcie funkcji tworzących do rozwiązywania rekurencji.