Teoretyczne podstawy informatyki

Wykład 10a:
O złożoności obliczeniowej raz jeszcze.

Złożoność zamortyzowana
Model danych zewnętrznych i algorytmy obróbki danych

Złożoność zamortyzowana

W wielu sytuacjach na strukturach danych działają nie pojedyncze operacje ale ich sekwencje. Jedna z operacji takiej sekwencji może wpływać na dane w sposób powodujący modyfikacje czasu wykonania innej operacji.
Jednym ze sposobów określania czasu wykonania w przypadku pesymistycznym dla całej sekwencji jest dodanie składników odpowiadających wykonywaniu poszczególnych operacji. Jednak wynik tak uzyskany może być zbyt duży w stosunku do rzeczywistego czasu wykonania. Analiza amortyzacji pozwala znaleść bliższą rzeczywistej złożoność średnią.

Analiza z amortyzacja polega na analizowaniu kosztów operacji, zaś pojedyncze operacje są analizowane właśnie jako elementy tego ciągu. Koszt wykonania operacji w sekwencji może być różny niż w przypadku pojedynczej operacji, ale ważna jest też częstość wykonywania operacji.

Jeśli dana jest sekwencja operacji op1, op2, op3,..., to analiza złożoności pesymistycznej daje daje złożoność obliczeniowa równa:

$C(op1, op2, op3,....) = C_{pes}(op1) + C_{pes}(op2) + C_{pes}(op3) + ...$

dla złożoności średniej uzyskujemy

$C(op1, op2, op3,....) = C_{sre}(op1) + C_{sre}(op2) + C_{sre}(op3) + ...$

Nie jest analizowana kolejność operacji, "sekwencja" to po prostu "zbiór" operacji.

Przy analizie z amortyzacją zmienia się sposób patrzenia, gdyż sprawdza się co się stało w danym momencie sekwencji i dopiero potem wyznacza się złożoność następnej operacji:

$C(op1, op2, op3,....) = C(op1) + C(op2) + C(op3) + ...$

gdzie C może być złożonością optymistyczna, średnią, pesymistyczna lub jeszcze inną - w zależności od tego co działo się wcześniej.

Znajdowanie złożoności zamortyzowanej tą metoda może być zanadto skomplikowane. Znajomość natury poszczególnych procesów oraz możliwych zmian struktur danych używane są do określenia funkcji C, którą można zastosować do każdej operacji w sekwencji. Funkcja jest tak wybieralna aby szybkie operacje były traktowane jak wolniejsze niż w rzeczywistości, zaś wolne jako szybsze. Sztuka robienia analizy amortyzacji polega na znalezieniu funkcji C; takiej która dociąży tanie operacje dostatecznie aby pokryć niedoszacowanie operacji kosztownych.

Przyklad:
dodawanie elementu do wektora zaimplementowego jako elastyczna tablica

Przypadek optymistyczny: wielkość wektora jest mniejsza od jego pojemności, dodanie elementu ogranicza się do wstawienia go do pierwszej wolnej komórki. Koszt dodania nowego elementu to $O(1)$.

Przypadek pesymistyczny: rozmiar jest równy pojemności, nie ma miejsca na nowe elementy. Konieczne jest zaalokowanie nowego obszaru pamięci, skopiowanie do niego dotychczasowych elementów i dopiero dodanie nowego. Koszt wynosi wówczas O(rozmiar (wektor)). Pojawia się nowy parametr, bo pojemność można zwiększać o więcej niż jedną komórkę wtedy przepełnienie pojawia się tylko "od czasu do czasu".

Analiza z amortyzacją: badane jest jaka jest oczekiwana wydajność szeregu kolejnych wstawień. Wiadomo, ż we przypadku optymistycznym jest to $O(1)$, w przypadku pesymistycznym O(rozmiar), ale przypadek pesymistyczny zdarza się rzadko.
Należy przyjąć pewna hipotezę:

a)

kosztAmort(push(x)) = 1

niczego nie zyskujemy, łatwe wstawienia nie wymagają poprawek, nie pojawia się jednak zapas na kosztowne wstawienia

b)

kosztAmort(push(x)) = 2

zyskuje zapas na łatwych wstawieniach, ale czy wystarczający…. zależy to od rozmiaru wektora….

rozmiar	pojemnosc	KosztAmort	koszt	zapas
0	0			
1	1	2	0+1	1
2	2	2	1+1	1
3	4	2	2+1	0
4	4	2	1	1
5	8	2	4+1	-2
6	8	2	1	-1
7	8	2	1	0
8	8	2	1	1
9	16	2	8+1	-6
10	16	2	1	-5
..	..	..	..	..
16	16	2	1	1
17	32	2	16+1	-14
18	32	2	1	-13

Operacje prawie cały czas są "na minusie" co jest niedopuszczalne

c)

kosztAmort(push(x)) = 3

zyskuje zapas na łatwych wstawieniach, ale czy wystarczający…. zależy to od rozmiaru wektora….

rozmiar	pojemnosc	KosztAmort	koszt	zapas
0	0			
1	1	3	0+1	2
2	2	3	1+1	3
3	4	3	2+1	3
4	4	3	1	5
5	8	3	4+1	3
6	8	3	1	5
7	8	3	1	7
8	8	3	1	9
9	16	3	8+1	3
10	16	3	1	5
..	..	..	..	..
16	16	3	1	17
17	32	3	16+1	3
18	32	3	1	5

Nigdy nie pojawia się "debet", zaoszczędzone jednostki są niemal w całości zużywane gdy pojawi się kosztowna operacja.

W przedstawionym przykładzie wybór funkcji stałej był słuszny. ale zwykle tak nie jest. Niech funkcja przypisująca liczbę do konkretnego stanu struktury danych ds będzie nazywana funkcją potencjału. Koszt amortyzowany definiuje się następująco:

$$koszAmort(op_i) = koszt(op_i) + f.potencjału(ds_i) - f.potencjału(ds_{i-1})$$

Jest to faktyczny koszt wykonania operacji op_i powiększony o zmianę potencjału struktury danych ds po wykonaniu tej operacji. Definicja ta obowiązuje dla pojedynczej operacji

$$koszAmort(op_1, op_2, op_3, \ldots, op_m) = \Sigma_{i=1}^{m} (koszt(op_i) + f.potencjału(ds_i) - f.potencjału(ds_{i-1})$$

W większości przypadków funkcja potencjału początkowo jest zerem, nigdzie nie jest ujemna, tak że czas amortyzowany stanowi kres górny czasu rzeczywistego.

kontunuacja przykładu:

$$f.potencjału(vector_i) = 0 \quad (\text{jeśli } rozmiar_i = pojemność_i \text{ czyli vector jest pełny})$$
$$= 2\, rozmiar_i - pojemność_i \quad (\text{w każdym innym przypadku})$$

Można sprawdzić że przy tak zdefiniowanej f.potencjału, $koszAmort(op_i)$ jest faktycznie równy 3 w każdej konfiguracji (tanie wstawianie, kosztowne, tanie po kosztownym)

Struktury danych i algorytmy obróbki danych zewnętrznych

Podstawowy czynnik rzutujący na różnice między obróbką danych wewnętrznych (czyli przechowywanych w pamięci operacyjnej) a obróbką danych zewnętrznych (czyli przechowywanych w pamięciach masowych) jest specyfika dostępu do informacji.

Mechaniczna struktura dysków sprawia, że korzystnie jest odczytywać dane nie pojedynczymi bajtami, lecz w większych blokach. Zawartość pliku dyskowego można traktować jako listę łączoną poszczególnych bloków, bądź też jako drzewo którego liście reprezentują właściwe dane, a węzły zawierają informację pomocniczą ułatwiającą zarządzanie tymi danymi.

Załóżmy że:

adres bloku = 4 bajty

długość bloku = 4096 bajtów

Czyli w jednym bloku można zapamiętać adresy do 1024 innych bloków.

Czyli informacja pomocnicza do 4 194 304 bajtów będzie zajmować 1 blok.

Możemy też budować strukturę wielopoziomową, w strukturze dwupoziomowej blok najwyższy zawiera adresy do 1024 bloków pośrednich, z których każdy zawiera adresy do 1024 bloków danych. Maksymalna wielkość pliku w tej strukturze 1024 * 1024 * 1024 = 4 294 967 296 bajtów = 4GB, informacja pomocnicza zajmuje 1025 bloków.

Nieodłącznym elementem współpracy pamięci zewnętrznej z pamięcią operacyjną są bufory, czyli zarezerwowany fragment pamięci operacyjnej, w której system operacyjny umieszcza odczytany z dysku blok danych lub z którego pobiera blok danych do zapisania na dysku.

Miara kosztu dla operacji na danych zewnętrznych.
Głównym składnikiem czasu jest czekanie na pojawienie się właściwego sektora pod głowicami. To może być nawet kilkanaście milisekund….. Co jest ogromnie długo dla procesora taktowanego kilku-gigahercowym zegarem.
Zatem "merytoryczna jakość" algorytmu operującego na danych zewnętrznych będzie zależna od liczby wykonywanych dostępów blokowych (odczyt lub zapis pojedynczego bloku nazywamy dostępem blokowym).

Sortowanie zewnętrzne to sortowanie danych przechowywanych na plikach zewnętrznych. Sortowanie przez łączenie pozwoli na posortowanie pliku zawierającego n recordow, przeglądając go jedynie O(log n) razy. Wykorzystanie pewnych mechanizmów systemu operacyjnego – dokonywanie odczytów i zapisów we właściwych momentach – może znacząco usprawnić sortowanie dzięki zrównoleglowieniu obliczeń z transmisją danych.

Sortowanie przez łączenie

Polega na organizowaniu sortowanego pliku w pewna liczbę serii, czyli uporządkowanych ciągów rekordów. W kolejnych przebiegach rozmiary serii wzrastają a ich liczba maleje; ostatecznie (posortowany) plik staje się pojedynczą serią.
Podstawowym krokiem sortowania przez łączenie dwóch plików, f1 i f2, jest zorganizowanie tych plików w serie o długości k, tak że:
-> liczby serii w plikach f1,f2, z uwzględnieniem "ogonów" różnią się co najwyżej o jeden
-> co najwyżej w jednym z plików f1, f2, może się znajdować ogon
-> plik zawierający "ogon" ma poza nim co najmniej tyle serii ile jego partner.
Prosty proces polega na odczytywaniu po jednej serii (o długości k) z plików f1, f2, łączenia tych serii w dwukrotnie dłuższą i zapisywania tak połączonych serii na przemian do plików g1, g2.
Całkowita liczba dostępów blokowych w całym procesie sortowania jest rzędu O((n logn)/b) gdzie b jest liczba rekordów w jednym bloku.

pliki orginalne

28 3 93 10 54 65 30 90 10 69 8 22

31 5 96 40 85 9 39 13 8 77 10

pliki zorganizowane w serie o dlugosci 2

28 31 93 96 54 85 30 39 8 10 8 10

3 5 10 40 9 65 13 90 96 77 22

pliki zorganizowane w serie o dlugosci 4

3 5 28 31 9 54 65 85 8 10 69 77

10 40 93 96 13 30 39 90 8 10 22

pliki zorganizowane w serie o dlugosci 8

3 5 10 28 31 40 93 96 8 8 10 10 22 69 77

9 13 30 39 54 65 85 90

pliki zorganizowane w serie o dlugosci 16

3 5 9 10 13 28 30 31 39 40 54 65 85 90 93 96

8 8 10 10 22 69 77

pliki zorganizowane w serie o dlugosci 32

3 5 8 8 9 10 13 22 28 30 31 39 40 54 65 69 77 85 90 93 96

Nie przejmuj się efektywnością algorytmu…. wystarczy poczekać kilka lat.

Taki pogląd funkcjonuje w środowisko programistów, nie określono przecież granicy rozwoju mocy obliczeniowych komputerów. Nie należy się jednak z nim zgadzać w ogólności. Należy zdecydowanie przeciwstawiać się przekonaniu o tym, ze ulepszenia sprzętowe uczynią prace nad efektywnymi algorytmami zbyteczna.

Istnieją problemy których rozwiązanie za pomocą zasobów komputerowych jest teoretycznie możliwe, ale praktycznie przekracza możliwości istniejących technologii. Przykładem takie problemu jest rozumienie języka naturalnego, przetwarzanie obrazów (do pewnego stopnia oczywiście) czy "inteligentna" komunikacja
Pomiędzy komputerami a ludźmi na rozmaitych poziomach.
Kiedy pewne problemy staja się "proste"…. Nowa grupa wyzwań, które na razie można sobie tylko próbować wyobrażać, wytyczy nowe granice możliwości wykorzystania komputerów.