SPRAWOZDANIE Z PROJEKTU: STRUKTURY BAZ DANYCH

Projekt nr 2

Autor: Krzysztof Taraszkiewicz **Wydział:** ETI **Kierunek:** informatyka **Data:** 06.12.2025

---

1. Cel i treść zadania

Celem projektu było zaprojektowanie i zaimplementowanie indeksowej organizacji pliku w oparciu o strukturę **B-Drzewa**. Zadanie obejmowało symulację obsługi pamięci zewnętrznej (dysku), realizację podstawowych operacji na danych (CRUD) oraz przeprowadzenie eksperymentów badających wydajność struktury.

Zrealizowane podzadania:

1. Symulacja dysku z podziałem na strony o stałym rozmiarze (512 bajtów).
2. Implementacja B-Drzewa z obsługą operacji: INSERT, SEARCH, UPDATE, DELETE.
3. Obsługa przepełnienia (split) i niedomiaru (merge/compensation) stron.
4. Implementacja mechanizmu ponownego wykorzystania zwalnianego miejsca (reusing space).
5. Interfejs testowy oraz moduł eksperymentalny.

---

2. Opis implementacji

2.1. Metoda indeksowania

Do realizacji zadania wybrano strukturę **B-Drzewa** (klasycznego).

- **Węzły drzewa** zawierają klucze , wskaźniki na strony potomne oraz wskaźniki (adresy stron) do właściwych rekordów danych .
- **Plik danych** jest oddzielony od pliku indeksu. Rekordy są przechowywane w strukturze stertowej (heap file), a indeks przechowuje jedynie ich adresy (Page ID).

2.2. Symulacja pamięci zewnętrznej

Zgodnie z wymaganiami, warstwa fizyczna została odseparowana od logiki drzewa.

- **Klasa DiskManager:** Operuje na pliku binarnym. Dzieli plik na bloki o rozmiarze PAGE_SIZE = 512 B.
- **Adresowanie:** Każda strona posiada unikalne ID. Strona o ID 0 jest zarezerwowana na metadane (np. wskaźnik na korzeń drzewa, licznik stron).

- **Serializacja:** Do zapisu obiektów węzłów i rekordów na binarne strony wykorzystano moduł pickle (z kontrolą rozmiaru, aby nie przekroczyć 512 bajtów). Zapewnia to symulację ograniczeń fizycznych dysku.

2.3. Zarządzanie buforowaniem i pamięcią

W implementacji zastosowano strategię **zapisu bezpośredniego (Write-Through)** bez skomplikowanego menedżera buforów (Buffer Pool) typu LRU.

- Każde żądanie dostępu do węzła lub rekordu powoduje odczyt ze "strony dyskowej" (pliku).
- Podczas operacji modyfikujących (np. split węzła), zmiany są natychmiast zapisywane na dysk, co zapewnia spójność danych, ale generuje większą liczbę operacji I/O.
- **Ograniczenia:** Ze względu na brak cache'owania w pamięci RAM, liczba odczytów w eksperymentach jest ściśle powiązana z wysokością drzewa.

2.4. Mechanizm zwalniania miejsca (Bonus)

Zaimplementowano mechanizm ponownego wykorzystania miejsca po usuniętych rekordach.

- Klasa DataFileManager utrzymuje w pamięci listę free_pages.
- Gdy rekord jest usuwany (DELETE), jego strona w pliku danych jest oznaczana jako wolna (zapisywane są same zera), a jej ID trafia do listy wolnych stron.
- Podczas operacji INSERT, system najpierw sprawdza listę free_pages. Jeśli jest niepusta, nadpisuje starą stronę nowym rekordem zamiast alokować nową na końcu pliku.

---

3. Specyfikacja formatu pliku testowego

Program umożliwia wczytywanie komend z pliku tekstowego (tryb skryptowy). Każda linia zawiera jedną instrukcję.

Format komend:

- **Dodawanie:** ADD <Klucz> <Liczba1> <Liczba2> ...
    - Przykład: ADD 105 12 44 5
- **Usuwanie:** DEL <Klucz>
    - Przykład: DEL 105
- **Aktualizacja:** UPD <Klucz> <NowaLiczba1> <NowaLiczba2> ...

- Przykład: UPD 105 99 88

- **Komentarze:** Linie zaczynające się od # są ignorowane.

---

4. Prezentacja wyników działania programu

Program działa w trybie interaktywnym oraz umożliwia wyświetlanie stanu struktur.

1. **Wyświetlanie Drzewa (print):** Prezentuje strukturę hierarchiczną indeksu, pokazując klucze w poszczególnych węzłach oraz ich relacje (rodzic-dziecko). Umożliwia weryfikację poprawności budowy B-Drzewa.

--- Struktura B-Drzewa ---

Strona 1: [30]

Strona 2: [10, 20]

Strona 3: [40, 50]

2. **Wyświetlanie Rekordów (scan):** Realizuje przegląd sekwencyjny (In-Order). Przechodzi przez indeks posortowany kluczami i dla każdego klucza pobiera odpowiedni rekord z pliku danych.

Klucz: 10 -> [ID: 10 | Liczby: [1, 2] | Suma: 3]

Klucz: 20 -> [ID: 20 | Liczby: [5, 5] | Suma: 10]

3. **Statystyki I/O:** Po każdej operacji program wyświetla liczbę wykonanych odczytów i zapisów stron dyskowych.

    - Przykład: IO: Odczyty: 4, Zapisy: 2

---

5. Eksperymenty

5.1. Metodyka

Przeprowadzono serię eksperymentów mających na celu zbadanie wpływu rzędu drzewa () oraz liczby rekordów () na wydajność operacji dyskowych i rozmiar plików.

- **Zmienne niezależne:**
    - Rząd drzewa .
    - Liczba rekordów .
- **Zmienne zależne (mierzone):**
    - Średnia liczba odczytów stron na jedną operację wstawiania.

- Średnia liczba zapisów stron na jedną operację wstawiania.
- Rozmiar pliku indeksu (w bajtach).

- **Przebieg:** Dla każdej pary wygenerowano losowe rekordy, wstawiono je do pustej bazy i zliczono operacje dyskowe.

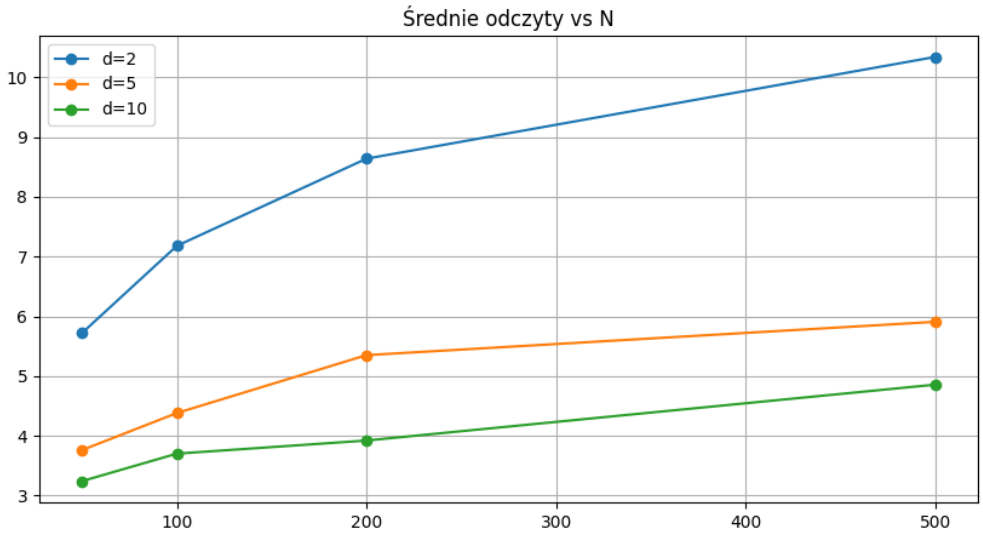
5.2. Wyniki eksperymentów

Tabela 1. Zestawienie wyników pomiarów (dla PAGE_SIZE = 512B)

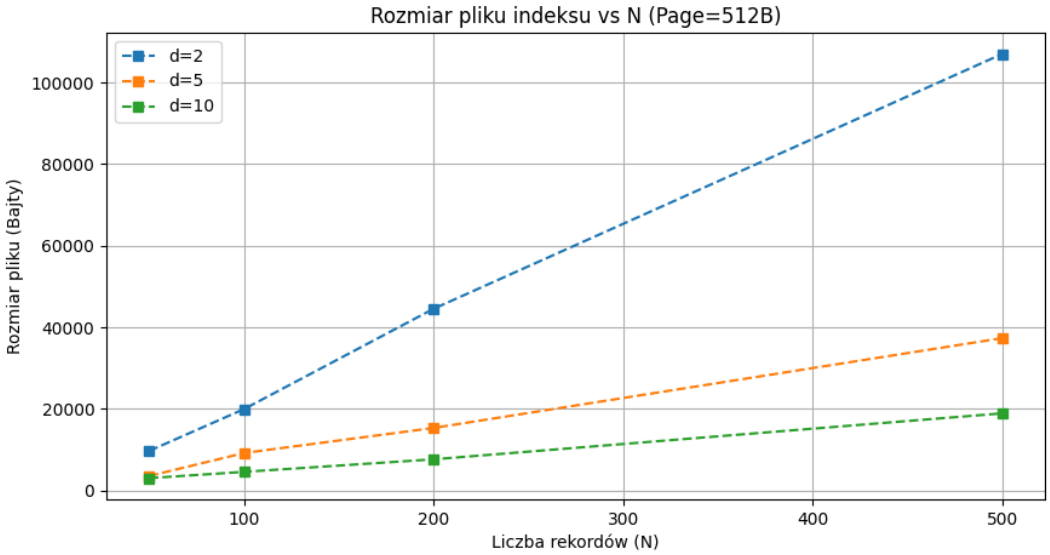
Rząd (d)	Liczba rekordów (N)	Śr. Odczyty	Śr. Zapisy	Rozmiar Indeksu (B)
2	50	5.72	4.20	9728
2	100	7.18	4.99	21504
2	200	8.64	5.76	46080
2	500	10.34	6.60	111616
5	50	3.76	3.00	4096
5	100	4.38	3.33	8192
5	200	5.35	3.82	15872
5	500	5.91	4.10	39424
10	50	3.24	2.68	2560
10	100	3.70	2.92	4608
10	200	3.92	3.03	8192
10	500	4.86	3.50	19456

5.3. Wykresy

Wykres 1. Średnia liczba odczytów dyskowych w funkcji liczby rekordów (N) dla różnych rzędów (d).



Wykres 2. Rozmiar pliku indeksu w funkcji liczby rekordów (N).



5.4. Analiza wyników i wnioski

1. **Wpływ rzędu drzewa () na odczyty:** Zgodnie z teorią, koszt wyszukiwania/wstawiania jest proporcjonalny do wysokości drzewa .
    - Dla małego , drzewo jest wyższe (wąskie), co skutkuje **większą** liczbą odczytów stron dyskowych, ponieważ musimy przejść przez więcej węzłów, aby dotrzeć do liścia.

- Dla dużego , drzewo jest niższe (szerokie), co drastycznie **zmniejsza** liczbę odczytów. Węzły mieszczą więcej kluczy, więc rzadziej schodzimy w dół struktury.

2. **Zapisy i podziały (Split):** Liczba zapisów jest skorelowana z częstotliwością podziałów stron. Przy większym , strony zapełniają się wolniej, co prowadzi do rzadszych operacji split, a tym samym mniejszej liczby zapisów strukturalnych.
3. **Rozmiar pliku indeksu:** Drzewa o wyższym rzędzie zazwyczaj lepiej utylizują miejsce w pliku (mniej pustych wskaźników i nagłówków stron w stosunku do danych), choć zależy to od stopnia wypełnienia węzłów.

---

6. Podsumowanie

Projekt pozwolił na praktyczne zapoznanie się z problematyką organizacji danych na dysku. Kluczowe wnioski z realizacji to:

- Zastosowanie B-Drzewa znacząco redukuje liczbę dostępów do dysku w porównaniu do struktur liniowych, zwłaszcza dla dużych .
- Wybór parametru jest kompromisem między wysokością drzewa (koszt odczytu) a czasem przetwarzania pojedynczego węzła (choć przy operacjach dyskowych czas CPU jest pomijalny).
- Mechanizm zwalniania miejsca ("Space Reuse") zapobiega niekontrolowanemu rozrostowi pliku danych przy intensywnych operacjach usuwania i dodawania rekordów.