1. Opisz tworzenie indeksu bitmapowego na przykładnie relacji OSOBY(IMIE,WIEK) o następujących danych:
(Alicja,17), (Beata,19), (A,19), (Cecylia,1), (Beata, 25), (Danuta,25), (Cecylia,19), (Alicja,25), (Edward,32), (Cecylia,18), (Danuta,19), (Edward,25), (Edward,31), (Beata,32), (Danuta,17)
Wykorzystując indeks zaprezentuj działanie operacji AND i OR w wyszukiwaniu rekordów dla warunku: (IMIE = 'Alicja' AND WIEK = 25) OR (IMIE = 'Danuta' AND WIEK = 19).

Alicja - 100000010000000
Beata - 010010000000010
A - 001000000000000
Cecylia- 000100100100000
Danuta- 000001000010001
Edward-000000001001100
17 - 100000000000001
19 - 011000100010000
1 - 000100000000000
25 - 000011010001000
32 - 000000001000010
18 - 000000000100000
31 - 000000000000100
Alicja - 100000010000000
25 - 000011010001000
AND - 000000010000000
Danuta- 000001000010001
19 - 011000100010000
AND - 000000000010000
AND1 - 000000010000000
AND2 - 000000000010000
OR - 000000010010000
ver 2
(A,7),(B,9),(A,9),(C,1),(B,2),(D,2),(C,9),(A,2),(E,3),(C,8),(D,9),(E,2)
(x=A AND y=9) OR (x=D AND y=2)
ver 3
(27,113), (45,113), (32,100), (32,75), (32,121), (30,121), (30,321), (45,140), (18,110), (50,275), (45,350), (27,400), (32,110)
(a=32 OR a=45) AND (b=110 OR b=113)

2. Opisz metodę kompresji i dekompresji przez kodowanie długości serii na przykładzie ciągu:
***ver 1* 00101000000010000000110100010000000000000000100001 .**
***ver 2* 00010000000000101000000110010000001000000010001**
***ver 3* 00000000001000101000000110000000000000000001**
wyznaczamy ciagi zer:
10 zer, 3 zera, 1 zero, 6 zer, 0 zer, 17 zer
zapisujemy kazda z tych liczb w postaci binarnej:
10 = 1010
3 = 11
1 = 1
6 = 110
0 = 0
17 = 10001

- 10 ma dlugosc 4 bity wiec w ciagu trzeba dopisac 1110(zero na 4 miejscu), wiec wynik bedzie 1110|1010
- 3 ma dlugosc 2 bity wiec w ciagu trzeba dopisac 10(zero na 2 miejscu), wiec wynik bedzie 10|11
- 1 ma dlugosc 1 bit wiec w ciagu trzeba dopisac 0(zero na 1 miejscu), wiec wynik bedzie 0|1
- 6 ma dlugosc 3 bity wiec w ciagu trzeba dopisac 110(zero na 3 miejscu), wiec wynik bedzie 110|110
- 0 ma dlugosc 1 bit w ciagu trzeba dopisac 0(zero na 1 miejscu), wiec wynik bedzie 0|0
- 17 ma dlugosc 5 bity wiec w ciagu trzeba dopisac 11110(zero na 5 miejscu), wiec wynik bedzie 11110|10001
  teraz trzeba polaczyc wszystkie wyniki:
  11101010|1011|01|110110|00|1111010001

3. Przy założeniu, że system działa w układzie RAID 5 z kodem korekcji Hamminga, wiedząc że dyski D1-D4 zawierają dane, a

dyski D5-D7 są redundancyjne i ich związek jest zdefiniowany za pomocą macierzy:
Dysk
1 2 3 4 | 5 6 7
----
1 1 1 0 | 1 0 0
1 1 0 1 | 0 1 0
1 0 1 1 | 0 0 1
Odtwórz brakujące bajty przy następującej sytuacji (po awarii)
D1 11110000
D2 ????????
D3 01111000
D4 01000001
D5 ????????
D6 10111110
D7 10001001
Odpowiedz uzasadnij opisując proces odtwarzania!

D1 11110000
D4 01000001
D6 10111110
---------- XOR
D2 00001111

D1 11110000
D2 00001111
D3 00111000
---------- XOR
D5 11000111
ver 2
Dysk
1 2 3 4 | 5 6 7
----
1 1 1 0 | 1 0 0
1 0 1 1 | 0 1 0
1 1 0 1 | 0 0 1
odtwórz brakujące bajty przy następującej sytuacji (po awarii):
D1: 11110000
D2: 01011010
D3: 00111010
D4: xxxxxxxx
D5: 10010000
D6: xxxxxxxx
D7: 11100011

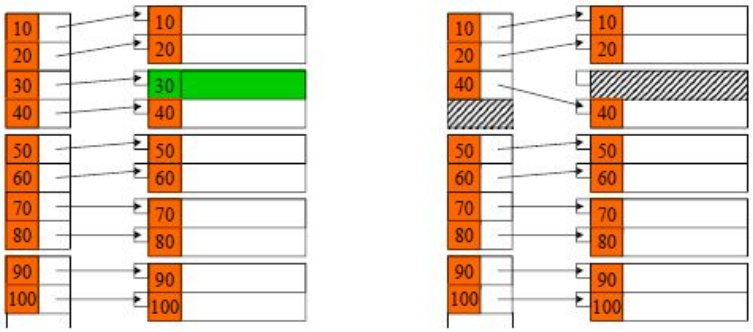
4. Opisz szczegółowo własności indeksów gęstych i rzadkich dla plików sekwencyjnych (podaj przykłady i opisz różnice). Przedstaw ideę indeksów pomocniczych.

- Indeks gęsty (*dense*) – zawiera wpis dla każdej wartości klucza wyszukiwania, czyli dla każdego rekordu.
- Indeks rzadki (*sparse*) – posiada wpis jedynie dla niektórych wartości wyszukiwania (np. bloków).

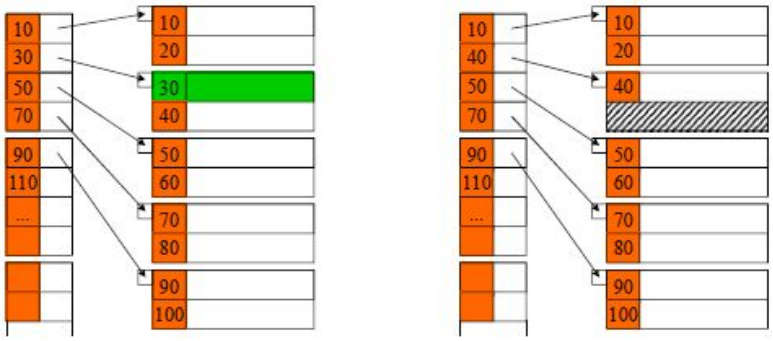
- w razie potrzeby tworzy się (lub usuwa) bloki nadmiarowe (wyłącznie jako poszerzenie bloków oryginalnych, czyli bez wskazań na nie w pliku indeksu rzadkiego)
- zamiast nadmiarowych możliwe jest tworzenie nowych bloków w pliku sekwencyjnym (z zachowaniem kolejności rekordów)
- w razie potrzeby rekordy przesuwane są między sąsiednimi blokami

	Gęsty	Rzadki
Tworzenie pustego bloku nadmiarowego	Brak	Brak
Usuwanie pustego bloku nadmiarowego	Brak	Brak
Tworzenie pustego bloku sekwencyjnego	Brak	Wstawianie
Usuwanie pustego bloku sekwencyjnego	Brak	Usuwanie
Wstawianie rekordu	Wstawianie	Aktualizacja
Usuwanie rekordu	Usuwanie	Aktualizacja
Przesuwanie rekordu	Aktualizacja	Aktualizacja

Przykład usuwania z pliku sekwencyjnego z indeksem gęstym (rekord o kluczu 30)



Przykład usuwania z pliku sekwencyjnego z indeksem rzadkim (rekord o kluczu 30)

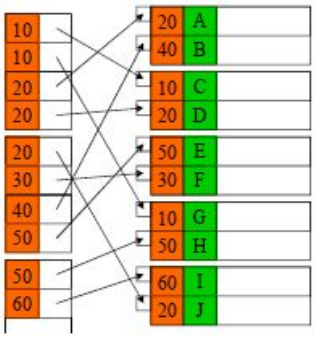


Indeksy pomocnicze – indeks zdefiniowany na podstawie pola, które nie jest używane przy wyznaczaniu porządku rekordów w bazie

- Dotychczasowe indeksy (**indeksy główne**) określały położenie rekordów.

SELECT nazwisko, adres FROM OSOBY
WHERE data_urodzenia = to_date('15.11.1974','DD.MM.YYYY');

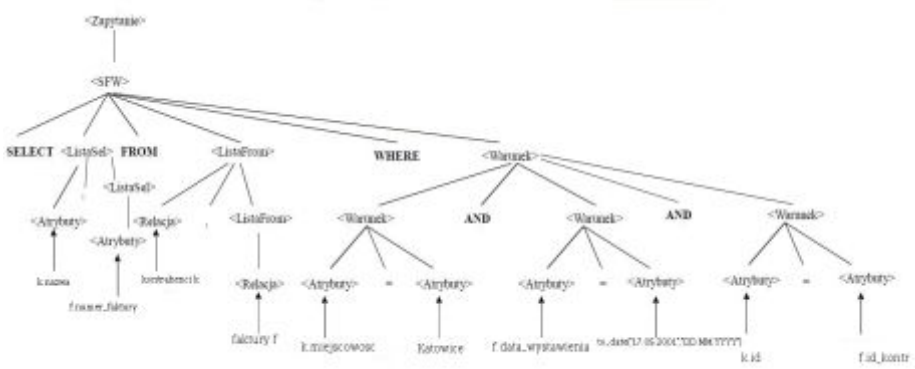
- Indeks główny na polu nazwisko nie wspiera zapytania z warunkiem na datę urodzenia;
- **Indeks pomocniczy** utworzony na polu data_urodzenia może wspierać takie zapytanie
- Indeks pomocniczy nie wyznacza pozycji rekordów w pliku z danymi, wskazuje jedynie ich bieżącą lokalizację.
- Nie ma więc sensu mówić o rzadkim indeksie pomocniczym (nie dałoby się ustalić wg takiego indeksu rekordów jawnie w nim nie występujących).
- Można stworzyć drugi poziom indeksu, który może już być rzadki.

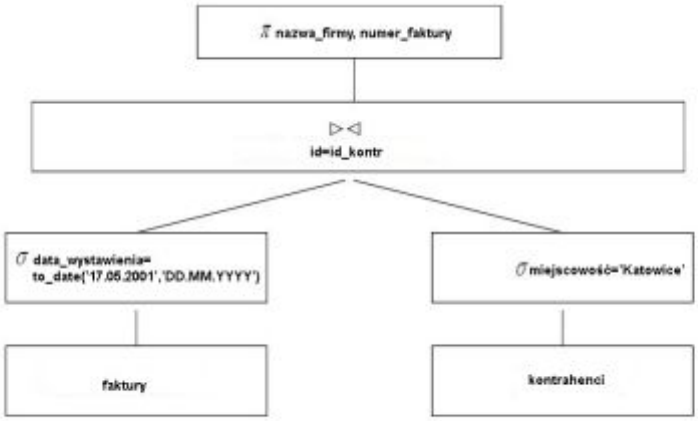


5. Podaj zasadę kosztu wejścia-wyjścia i przedstaw ideę algorytmu dwufazowego wielowejściowego sortowania przez scalenie.
Jeżeli zachodzi konieczność przemieszczenia bloku danych między dyskiem a pamięcią operacyjną to czas potrzebny do odczytu bądź zapisu jest znacznie większy, niż czas potrzebny na przetworzenie danych w pamięci operacyjnej. Dlatego liczba bloków, do których należy zapewnić dostęp (odczytu lub zapisu) jest dobrym przybliżeniem czasu wykonania algorytmu i te liczbę należy minimalizować.

Dwufazowe wielowejściowe sortowanie przez scalanie
Faza 1: Posortuj fragmenty danych, które mieszczą się w pamięci operacyjnej, tak aby każdy rekord znalazł się na dokładnie jednej posortowanej liści mieszczącej się w pamięci operacyjnej (takich podlist może być wiele)
Faza 2: Wszystkie posortowane listy z fazy pierwszej scal w jedną listę.
W fazie pierwszej:
- cała dostępna pamięć operacyjna jest zapełniona blokami danych z relacji, której rekordy mają być posortowane
- rekordy wczytane do pamięci są sortowane
- rekordy posortowane zapisywane są z pamięci do nowych bloków pamięci pomocniczej, tworząc jedną posortowaną podlistę.
Po zakończeniu pierwszej fazy każdy rekord oryginalnej relacji był wczytany jeden raz do pamięci, stał się częścią jednej posortowanej listy mieszczącej się w pamięci, która to lista została zapisana na dysk.
Inne pytania:
5. Wykorzystując symbolikę algebry zapytań dla relacji:
kontrahenci (id, nazwa_firmy, miejscowość)
faktury (numer_faktury, id_kontr, data_wystawienia)
Narysuj drzewo wyrażeń zapytania:
Kod:

```
SELECT k.nazwa_firmy, f.numer_faktury
FROM kontrahenci k, faktury f
WHERE
k.miejscowość='Katowice'
AND f.data_wystawienia=to_date('17.05.2001','DD.MM.YYYY')
AND k.id=f.id_kont
```


<Zapytanie>
<SFW>
SELECT
<ListaSel>
FROM
<ListaFrom>
WHERE
<Warunek>
<Atrybuty>
<ListaSel>
<Atrybuty>
<Relacja>
<ListaFrom>
<Warunek>
AND
<Warunek>
AND
<Warunek>
k.nazwa
f.numer_faktury
kontrahenci k
<Relacja>
<Atrybuty>
=
<Atrybuty>
<Atrybuty>
=
<Atrybuty>
<Atrybuty>
=
<Atrybuty>
faktury f
k.miejscowosc
Katowice
f.data_wystawienia
k.id
f.id_kontr


π nazwa_firmy, numer_faktury
⋈
id=id_kontr
σ data_wystawienia= to_date('17.05.2001','DD.MM.YYYY')
σ miejscowość='Katowice'
faktury
kontrahenci

1. Scharakteryzuj pamięci występujące w SZBD.
Hierarchia pamięci: podręczna, operacyjna, wirtualna, pomocnicza, trzeciego poziomu.

Pamięć podręczna:
- zintegrowany układ scalony (lub fragment procesora)
- przechowuje dane i instrukcje, zwykle w niewielkich ilościach, które trafiają z określonego miejsca pamięci operacyjnej
- pamięć podręczna jest czasem dzielona na pamięć podręczną płyty i pamięć podręczną drugiego poziomu
- Gdy instrukcje są wykonywane, wówczas sam rozkaz i dane są pobierane do pamięci podręcznej. Jeśli nie można ich tam odnaleźć, to są kopiowane z PAO. Często dane zawarte w pamięci podręcznej muszą być zamieniane nowymi. Jeśli nie były zmieniane, to zostają po prostu zastąpione następnymi, jeśli były na nich wykonywane zmiany, to muszą wcześniej zostać skopiowane do PAO.
- W systemach wieloprocesorowych, gdzie kilka procesorów korzysta z tej samej PAO, często dane z pamięci podręcznej muszą być natychmiast odwzorowywane w pamięci operacyjnej.
- pojemność pamięci podręcznej: ok. 1MB
- czas wymiany z procesorem lub wykonania rozkazu: 10^8
- czas wymiany z pamięcią operacyjną: 10^-7

Pamięć operacyjna:
- O wszelkich operacjach wykonywanych w komputerze można myśleć jak o operacjach wykonywanych w pamięci operacyjnej, gdyż wymiana z pamięcią podręczną jest tak szybka, że można ją pominąć przy rozważaniach związanych z czasem.
- pamięć operacyjna charakteryzuje dostęp swobodny, tzn. każdy bajt jest dostępny w takim samym czasie
- pojemność pamięci operacyjnej: 100MB - 10GB i więcej)
- czas dostępu do danych: 10 - 100 nanosekund

Pamięć wirtualna:
- program wykonywany przez komputer znajduje się w przestrzeni adresowej pamięci wirtualnej
- 32 bitowa przestrzeń adresowa pozwala zaadresować 2^32 adresów (ponad 4 miliardy adresów). Typowa pamięć wirtualna ma więc 4 GB.
- Jest to więcej niż dostępna pamięć operacyjna, zatem większość pamięci wirtualnej jest przechowywana na dysku.
- bloki dysku 4KB do 56 KB
- pamięć wirtualna jest przekazywana między dyskiem a pamięcią operacyjną w blokach nazywanych stronami

Pamięć pomocnicza:
- dyski (zwykle magnetyczne, optyczne i magnetooptyczne)
- pamięć wolniejsza, ale znacznie pojemniejsza, niż pamięć operacyjna
- czas dostępu do danych nie jest stały, ale różnice nie są duże. Dyski są więc podstawą dla pamięci pomocniczej, ale także dla pamięci wirtualnej
- DBMS'y często nie korzystają z pośrednictwa menadżera plików systemu operacyjnego, ale same organizują sobie wymianę między dyskiem i pamięcią operacyjną. Zasady pozostają jednak takie same.
- operacje zapisu i odczytu trwają w przybliżeniu 10-30 milisekund (0,01 - 0.03 s)
- W tym czasie przeciętna maszyna może wykonać milion rozkazów, dlatego tak istotna jest gospodarka pamięcią. W miarę możliwości jak najczęściej blok z danymi zapisany na dysku powinien znajdować się już w pamięci, gdy będzie potrzebny.

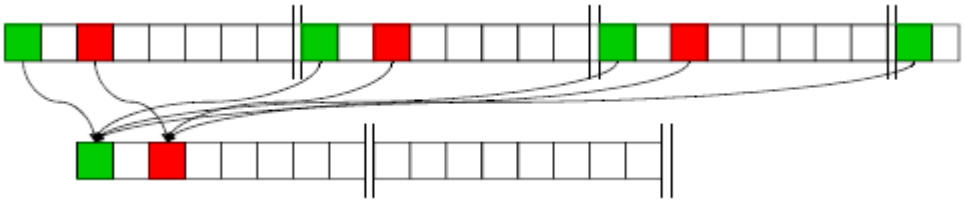
Pamięć III- go poziomu
- Stosowane są pamięci taśmowe, urządzenia automatycznie zmieniające dyski optyczne (jukebox), silosy taśmowe (robot przenosi taśmy do czytników).
- Pamięć III-go poziomu charakteryzuje się znacznie dłuższymi czasami dostępu i znacznie większymi pojemnościami.
- Pojemności rzędu terabajtów (jedna taśma potrafi przechowywać 50GB)
- czas dostępu: kilka sekund do kilku minut

3. Opisz typy awarii dysków, funkcjonowania sum kontrolnych i przechowywania stabilnego.
Typy awarii:
- zakłócenia sporadyczne - jedna lub kilka nieudanych prób zapisu lub odczytu sektora, po których zapis/ odczyt zostaje wykonany poprawnie

- uszkodzenia nośnika - trwałe uszkodzenie jednego lub większej ilości bitów (poprawne odczytanie sektora staje się niemożliwe)
- błąd zapisu - nie udaje się zapisać sektora, ani odczytać sektora zapisanego poprzednio
- uszkodzenie dysku - cały dysk niemożliwy do odczytu

Sumy kontrolne

- każdy sektor zawiera dodatkowe bity (sumę kontrolną), których wartość zależy od bitów danych pamiętanych w sektorze. Przy niezgodności status odczytu określa się jako niepoprawny.
- prawdopodobieństwo trafienia w dobrą sumę kontrolną mimo błędnych odczytów jest niewielkie, ale istnieje
- postać sumy kontrolnej nazuje na parzystości:
* jeśli w zbiorze bitów sektora jest nieparzysta liczba jedynek, to "parzystość jest ujemna" (lub "bit parzystości wynosi 1")
* w przeciwnym wypadku bity mają "dodatnią parzystość" (lub "bit parzystości wynosi 0")
- W efekcie liczba jedynek zapisywanych do sektora zawsze jest parzysta i bit parzystości jest zawsze dodatni
01101000 -> 01101000 **1**
11101110 -> 11101110 **0**
- każdy błąd przy zapisie lub odczycie bitu powoduje powstanie parzystości ujemnej
- sterownik sprawdza "w locie" liczbę jedynek w sektorze
- co się dzieje, że jeśli w sektorze bitów uszkodzonych jest więcej?
11101110 -> 11001100 **0**
- Można zwiększyć szansę wykrycia przez większą liczbę bitów parzystości.
- Na przykład utrzymywanie 8 bitów parzystości (po jednym dla każdego pierwszego, drugiego, trzeciego... bitu w bajtach)
- Prawdopodobieństwo niewykrycia tego, że wystąpił błąd wynosi tylko 1 przez 2^8
- Ogólnie - używając n bitów do kontroli parzystości otrzymujemy 1 / 2^n szans na niezauważenie błędu.
- Gdyby stosować na każdy sektor 3 bajty kontrolne są to szanse 1 : 4



miliardy

Przechowywanie stabilne:

- Sama świadomość faktu błędu w odczycie/zapisie to za mało - potrzebna jest możliwość przeciwdziałania.
- **Przechowywanie stabilne** polega na kompletowaniu par sektorów. Każda para jest przeznaczona do przechowywania zawartości X, która można zmieścić w jednym sektorze.
- Oznaczymy X_L i X_R - „lewą” i „prawą” kopię wartości X.
- Zakładamy, że kopie zapisujemy z taką kontrolą parzystości (na wystarczającej ilości bitów), że możemy wyeliminować ryzyko, że zły sektor wygląda jak dobry.
- strategia zapisu:
* zapisać X do X_L
* sprawdzić stan
* jeśli jest poprawny, to w kopii X_L bity parzystości są poprawne
- Jeśli nie - powtórzyć zapis.
* Jeśli po serii prób nie udało się poprawnie zapisać X do X_L to przyjmujemy, że w X_L wystąpiło uszkodzenie nośnika. Wtedy trzeba dokonać naprawy przez zastąpienie uszkodzonego sektora innym wolnym obszarem dysku.
* Wykonać wszystkie czynności dla X_R.
- **Strategia odczytu:**
***** Odczytać X_L jako wartość X.
***** Jeśli stan jest niepoprawny, to powtórzyć kilka razy.
***** jeśli się uda, to stan poprawny,
***** jeśli nie to wykonać to samo dla X_R.

4. Na czym polega różnica miedzy macierzami RAID 1,4,5,6. Podaj stosowne przykłady.

- system **RAID 1 -** najprostsza metoda (opisana wcześniej przy technikach poprawiania czasu dostępu do dysku jako **lustrzane kopiowanie danych**). W tym podejściu nie ma rozróżnienia na to, który dysk jest z danymi, a który redundancyjny. Ilość dysków redundancyjnych jest równa ilości dysków z

danymi, lustrzane kopiowanie danych, dyski redundancyjne to idealne kopie dysków z danymi; ilość dysków redundancyjnych równa się ilości dysków z danymi; odtworzenie danych to po prostu skopiowanie danych z dysku redundancyjnego; sposób najprostszy i najwolniejszy;
- system **RAID 4** - korzysta tylko z jednego dysku redundancyjnego bez względu na ilość dysków z danymi, dysk redundancyjny jest tylko jeden dla dowolnej ilości dysków z danymi; dane na dysku redundancyjnym to 'suma modulo 2' bajtów z dysków z danymi; odtworzenie danych to ponowne przeprowadzenie 'sumy modulo 2' (działa to zarówno dla dysków z danymi jak i dla dysku redundancyjnego); zawodzi przy awarii więcej niż jednego dysku;
- system **RAID 5** - brak jednoznacznego dysku redundancyjnego, sumy parzystości są rozpraszane po całej macierzy, nie chroni przed awarią większej ilości dysków. podobne założenia co w RAID 4, z tym że dysk redundancyjny nie jest określony i sumy kontrolne danych zapisywane są na każdym z dysków; dzięki temu przy zapisywaniu/zmienianiu danych nie trzeba wykonywać operacji na dysku redundancyjnym, przez co obciążenie zapisami/odczytami się wyrównuje (szybszy od RAID 4); odtworzenie działa tak samo jak w RAID 4;
- system **RAID 6** - często zapisywana jako RAID 5+1 ponieważ tak jak w RAID 5 nie ma jednego stałego dysku redundancyjnego; oprócz tego zamiast jednej są 2 sumy kontrolne co oznacza odporność na awarię dwóch dysków; szybszy niż RAID 5.

Przykład:
Pierwsze bajty bloków na dyskach z danymi:
D1: 11110000
D2: 10101010
D3: 00111000
Wówczas na redundacyjnym:
D4: 01100010 (suma modulo 2)
- Realizowane jest to w sposób szybszy, dzięki własnościom dodawania modulo 2, przez:
* Obliczenie sumy starej i nowej wartości zmienianego bloku
* Wykonanie zmian na odpowiednich pozycjach dysku redundancyjnego
D1: 11110000
D2: 10101010
D3: 00111000
* zmiana w bloku na dysku D2: 10101010 -> D2': 11001100
D2: 10101010
D2': 11001100

0**11**00**11**0
* odczyt dysku redundancyjnego
D4: 01100010
i zmiana na wartość przeciwną tych pozycji, na których suma D2 i D2' miała jedynkę
-> D4': 0**00**00**10**0
* Sprawdźmy:
D1: 11110000
D2': 11001100
D3: 00111000

D4': 00000100