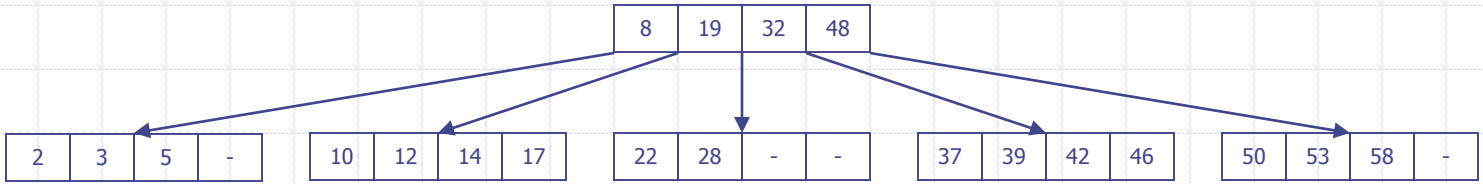
Arbori B

- Arborii B formeaza o categorie speciala de arbori, care se caracterizeaza in principal prin faptul ca un nod contine ***mai multe chei***, nu una singura
- Un astfel de nod care contine ***mai multe chei*** poarta denumirea de ***pagina***
- In cadrul arborilor B exista ***un mecanism de ordonare*** a paginilor oarecum similar cu cel de la arbori binari ordonati, ceea ce face ca o anumita pagina sa fie gasita foarte usor
- In cadrul unei pagini, cheile sunt dispuse, de regula, sub forma unui ***tablou ordonat*** ceea ce face ca o anumita cheie sa fie gasita foarte usor cu ajutorul unei cautari binare

Arbori B

- Iata un exemplu de arbore B:



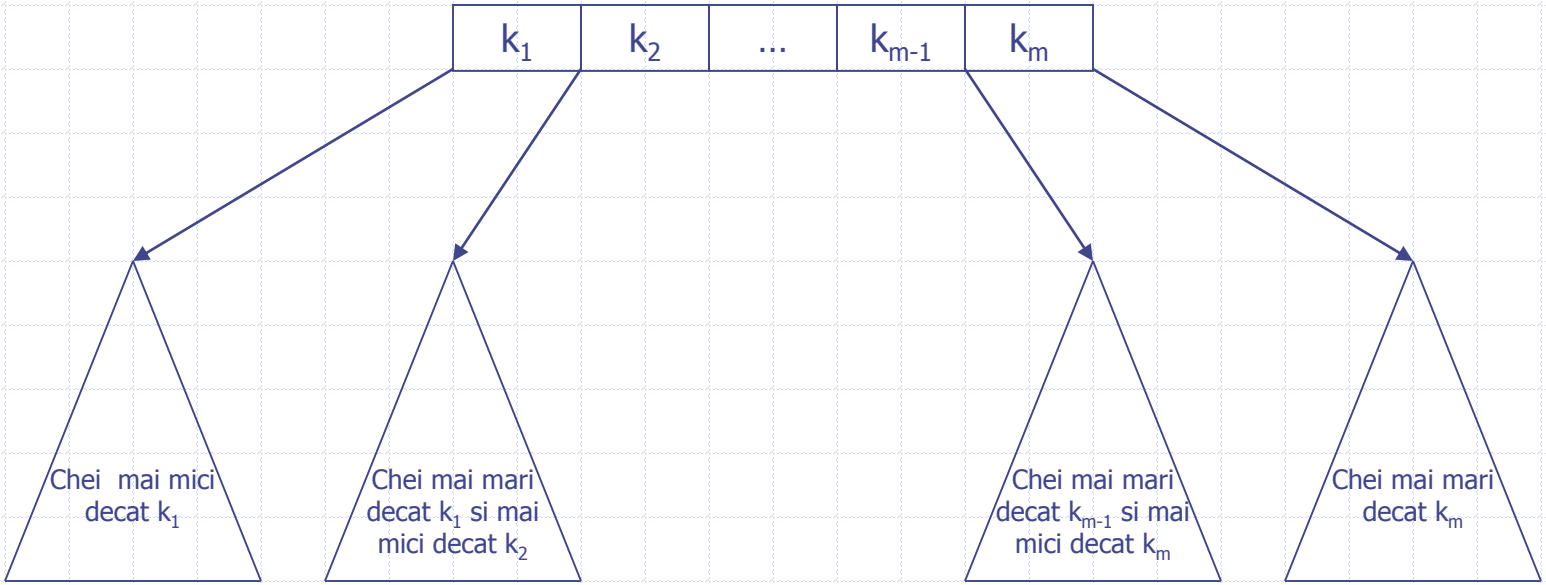
- Se observa ca:
  - In cadrul unei pagini, cheile sunt pastrate intr-un tablou ordonat
  - Numarul maxim de chei dintr-o pagina este fix (4, in cazul nostru)
  - Numarul maxim de fii ai unei pagini neterminale este de asemenea fix (5 = 4 + 1, in cazul nostru)

Arbori B

- O importanta majora o prezinta criteriul de ordonare a paginilor
- Acesta este oarecum similar cu cel de la arbori binari ordonati
- Fie pagina radacina (8, 19, 32, 48) din arborele precedent
- Subarborele stang al lui 8 contine numai chei mai mici decat 8 (2, 3, 5)
- Subarborele drept al lui 8 respectiv stang al lui 19 contine numai chei mai mari decat 8 si mai mici decat 19 (10, 12, 14, 17)
- Subarborele drept al lui 19 respectiv stang al lui 32 contine numai chei mai mari decat 19 si mai mici decat 32 (22, 28)
- Subarborele drept al lui 32 respectiv stang al lui 48 contine numai chei mai mari decat 32 si mai mici decat 48 (37, 39, 42, 46)
- Subarborele drept al lui 48 contine numai chei mai mari decat 48 (50, 53, 58)

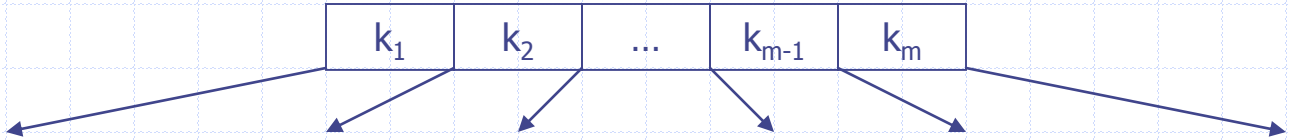
Arbori B

- In general, orice pagina neterminala intr-un arbore B este de forma:



Arbori B

- Principiul cautarii unei chei x intr-un arbore B este urmatorul:



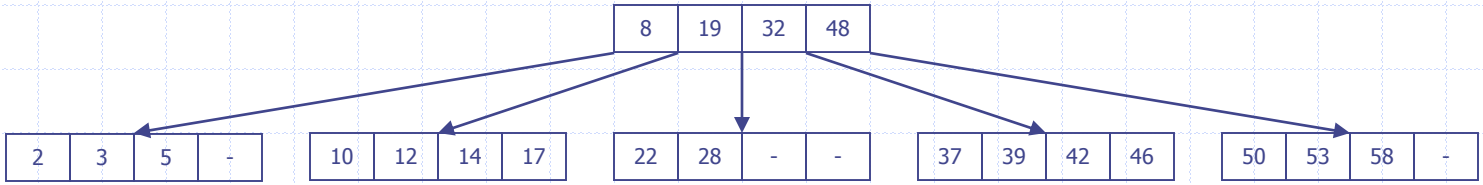
- Se cauta cheia x in pagina radacina (k_1, k_2, ..., k_m) folosind cautarea binara, deoarece cheile k_1, k_2, ..., k_m sunt pastrate ordonate
- Daca nu se gaseste, atunci in functie de relatia dintre cheia x si cheile paginii radacina se continua cautarea recursiv:
  - in subarborele nr. 1, daca $x < k_1$
  - in subarborele nr. 2, daca $x > k_1$ si $x < k_2$
  - ...
  - in subarborele nr. m+1, daca $x > k_m$
- La fiecare pas se continua cautarea intr-un singur subarbore si se exclud restul de m subarbori, ceea ce face ca procesul de cautare a unei pagini sa fie rapid convergent

Arbori B

- Orice arbore B are un ordin, notat cu N
- Ordinul unui arbore B controleaza numarul de chei care pot coexista intr-o pagina
- Astfel, un arbore B de ordinul N se caracterizeaza prin:
  - orice pagina contine **cel mult** 2·N chei
  - orice pagina contine **cel putin** N chei (cu exceptia paginii radacina, care ***poate contine si mai putin de N chei***)
  - orice pagina este sau pagina terminala (nu are descendenti) sau pagina neterminala, caz in care are m+1 descendenti (unde m este numarul de chei din pagina)
  - paginile terminale se afla **toate** pe acelasi nivel, deci inaltimea unui arbore B este data de lungimea **oricarui** drum de la radacina la o pagina terminala

Arbori B

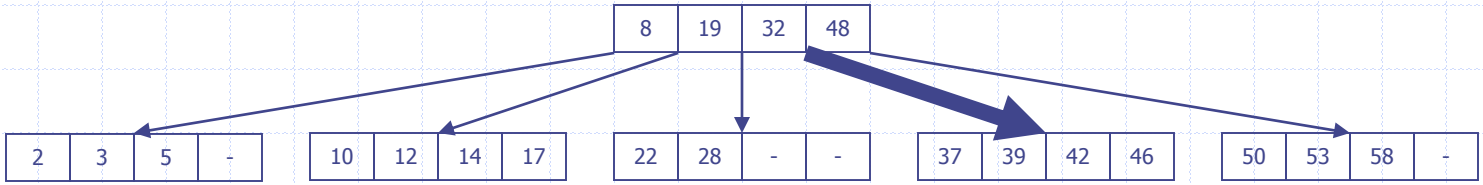
- Fie arborele B de ordinul 2 din figura:



- Fiecare pagina contine cel mult 4 (2·2) chei
- Fiecare pagina contine cel putin 2 chei (radacina poate fi exceptie de la aceasta regula, adica ar putea contine chiar o singura cheie, dar nu este cazul aici)
- Toate paginile terminale se afla pe acelasi nivel – aceasta conditie este asigurata implicit de algoritmii de insertie, respectiv suprimare de chei

Arbori B

- Cautarea unei chei (42, de exemplu) intr-un astfel de arbore se face simplu



- Se pleaca de la radacina si se cauta cheia 42 intre cheile radacinii
- Nu se gaseste, dar se observa ca 42 se situeaza ca valoare intre 32 si 48
- Cautarea va continua recursiv pe subarborele spre care indica sageata ingrosata de pe figura (sageata “dintre” 32 si 48)
- Cautarea in oricare alt subarbore este inutila din motive evidente

Arbori B

- Atat in arbori binari ordonati cat si in tablouri ordonate, cautarea se poate face intr-un timp proportional cu $\log_2 n$, unde n reprezinta numarul total de chei
- Arborii binari ordonati prezinta avantajul binecunoscut al structurilor dinamice, acela ca se aloca exact atat spatiu cat trebuie, dar accesul la informatie este mai greoi, deoarece se face, de obicei, prin pointeri (practic, se face un acces la memorie pentru valoarea pointerului si inca un acces la memorie pentru valoarea de la adresa pointerului)
- Tablourile ordonate prezinta avantajul accesului rapid (direct) la informatie, dar de obicei necesita alocarea unui spatiu de memorie mai mare decat este nevoie (se poate observa in arborele B dat ca exemplu ca exista pagini din care lipsesc chei, lucru care se datoreaza folosirii tablourilor)
- Arborii B incearca sa combine avantajele structurilor dinamice de cautare (arbori binari ordonati) cu avantajele structurilor statice (tablouri ordonate), maximizand avantajele si minimizand dezavantajele

Arbori B

- Cautarea unei chei intr-un arbore B de ordinul N presupune 2 operatii:
  - cautarea paginii in care poate fi gasita cheia – aceasta operatie se face rapid, deoarece dintr-o pagina cu m chei, cautarea continua pe subarborele corespunzator cheii care se cauta, neglijandu-se ceilalti m subarbori
  - odata gasita pagina, cautarea cheii in pagina respectiva – aceasta operatie se face de asemenea rapid, deoarece tabloul de chei din pagina este ordonat, asa ca se poate folosi un algoritm de cautare binara avand performanta logaritmica
- Aceste considerente fac ca arborii B sa fie structuri de date deosebit de performante
- Exista implementari de arbori B care nu folosesc deloc tablouri, cheile dintr-o pagina fiind la randul lor inlantuite intr-o lista