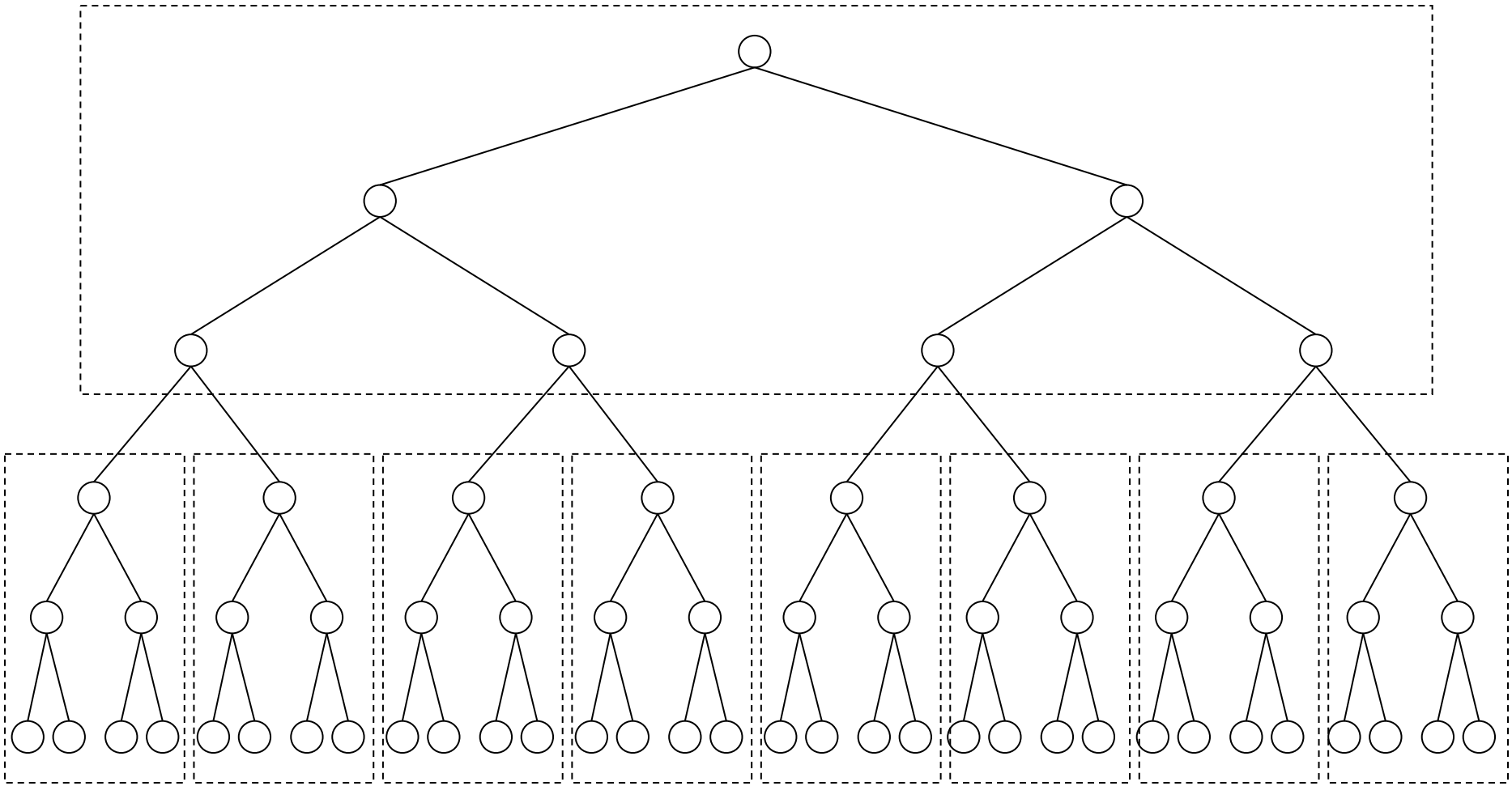
B – cây

- Cây nhị phân đáp ứng khá đầy đủ các yêu cầu về biểu diễn cấu trúc dữ liệu
- Trong thực tế, kích thước dữ liệu thường là lớn hoặc vô cùng lớn. Đòi hỏi phải lưu trữ ở bộ nhớ ngoài
  - Dung lượng “vô hạn”
  - Tốc độ chậm hơn khoảng 10^5 lần so với bộ nhớ trong, ngoài ra còn là chi phí di chuyển đầu đọc

- Sử dụng *cây nhiều nhánh* (lưu trữ trên đĩa) để biểu diễn cây tìm kiếm cỡ lớn
- Nhằm hạn chế tối đa thao tác truy xuất (đọc/ghi đĩa), cây được thiết kế với:
  - Đơn vị truy xuất là nhóm dữ liệu, gọi là *trang*
  - Chiều cao cây đạt mức tối thiểu

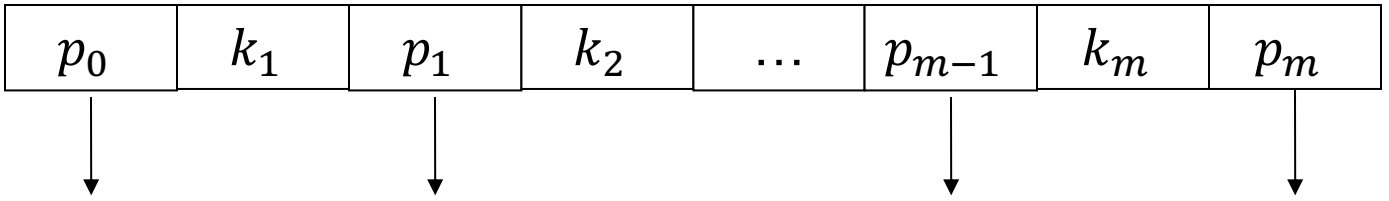
$\rightarrow$ Hình thành khái niệm B-cây

- Một trang tương ứng một nút của B-cây
- Mỗi trang ngoài việc lưu trữ dữ liệu còn chứa "liên kết" để quản lý các trang con

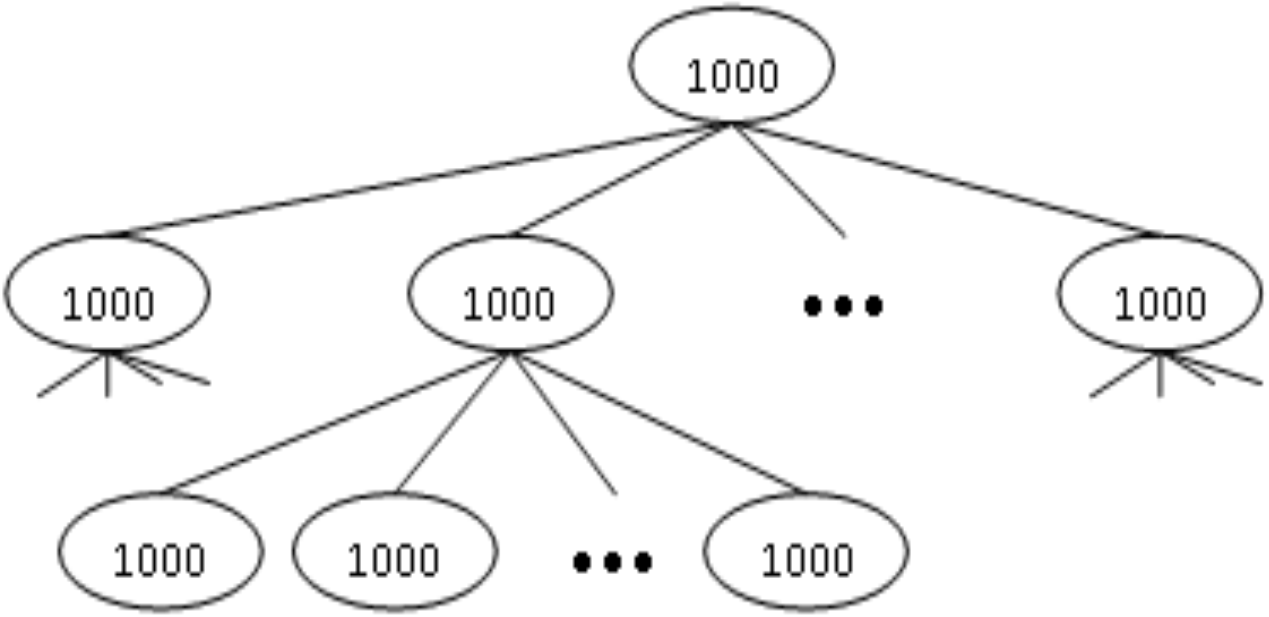


Định nghĩa: B-cây (bậc n) là cây cỡ lớn, được lưu trên đĩa với tổ chức như sau:

- Mỗi trang chứa tối đa $2n$ khóa
- Mỗi trang, trừ trang gốc, chứa tối thiểu n khóa
- Mỗi trang, trừ trang lá, nếu có m khóa thì có $m + 1$ trang con
- Mọi trang lá đều xuất hiện ở cùng mức
- Các khóa được sắp xếp tăng dần từ trái qua phải



Xét B-cây với số khóa trung bình là 1000



1 nút
(1000 khóa)

1001 nút
(1000 × 1001 =
1,001,000 khóa)

1001 × 1001 =1002001 nút
(1000 × 1002001 =
1,002,001,000 khóa)

- Với chiều cao là 3, cây chứa hơn 1 tỉ khóa (tương ứng với hơn 1 tỉ đơn vị dữ liệu)
- Nút gốc thường được lưu trong bộ nhớ chính nên chỉ mất tối đa 2 phép truy xuất để tìm một khóa

Tìm kiếm trên B-cây

1. Đọc trang vào bộ nhớ và tìm kiếm tuần tự
   - Nếu m đủ lớn thì dùng tìm kiếm nhị phân
2. Nếu không tìm thấy (gọi k là khóa cần tìm):

   a) $k_i < k < k_{i+1}$ với $1 \leq i < m$: tìm tiếp trên trang trỏ bởi p_i

   b) $k_m < k$: tìm tiếp trên trang trỏ bởi p_m

   c) $k < k_1$: tìm tiếp trên trang trỏ bởi p_0
3. Nếu con trỏ trỏ đến trang mới bằng NULL: không tìm thấy

Thêm phần tử vào B-cây

1. Tìm trang lá thích hợp và chèn vào đúng vị trí
2. Nếu trang đầy: tiến hành tách trang
   - Mỗi trang chứa n khóa, phần tử đứng giữa được đưa lên trang cha
3. Trường hợp trang cha cũng trở nên đầy: Tiến trình tiếp tục theo chiều đi lên

   $\rightarrow$ Cây lớn lên từ lá đến gốc

Xóa phần tử trên B-cây

Giai đoạn 1: Tìm và xóa phần tử

- ở trang lá: xóa bình thường
- ở trang trong:
  - Tìm *phần tử thay thế* (ở trang lá)
    - Đi theo con trỏ phải nhất của trang con bên trái, hoặc
    - Đi theo con trỏ trái nhất của trang con bên phải
  - Sao chép dữ liệu của *phần tử thay thế* cho phần tử định xóa ban đầu
  - Xóa *phần tử thay thế*

Giai đoạn 2: Kiểm tra "tính cân bằng"

- Việc xóa làm giảm số phần tử m của trang lá đi 1 đơn vị
- Nếu $m \geq n$: Dừng
- Nếu $m < n$: Xét *trang kế bên*

  $> n$ phần tử: Cân bằng số lượng phần tử của 2 trang (thông qua phần tử trung gian ở trang cha)

  $= n$ phần tử: Kéo phần tử trung gian ở trang cha xuống và ghép 2 trang $\rightarrow$ trang đầy

  Quá trình có thể lan truyền ngược … (nếu trang cha trở nên thiếu) … về đến gốc $\rightarrow$ cây giảm chiều cao