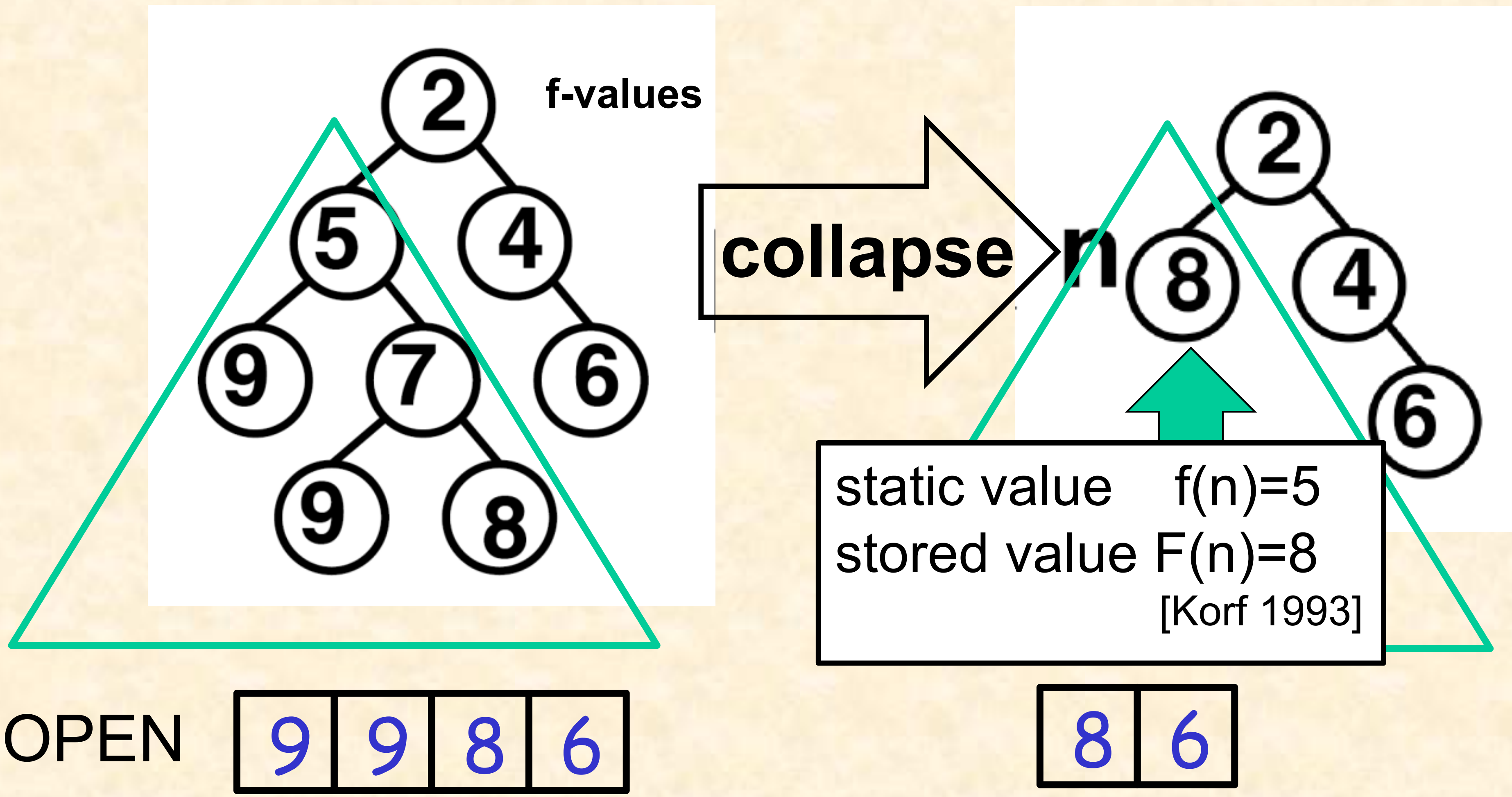
Search for Optimal Solutions: the Heart of Heuristic Search is Still Beating

Ariel Felner
ISE Department
Ben-Gurion University
ISRAEL
felner@bgu.ac.il

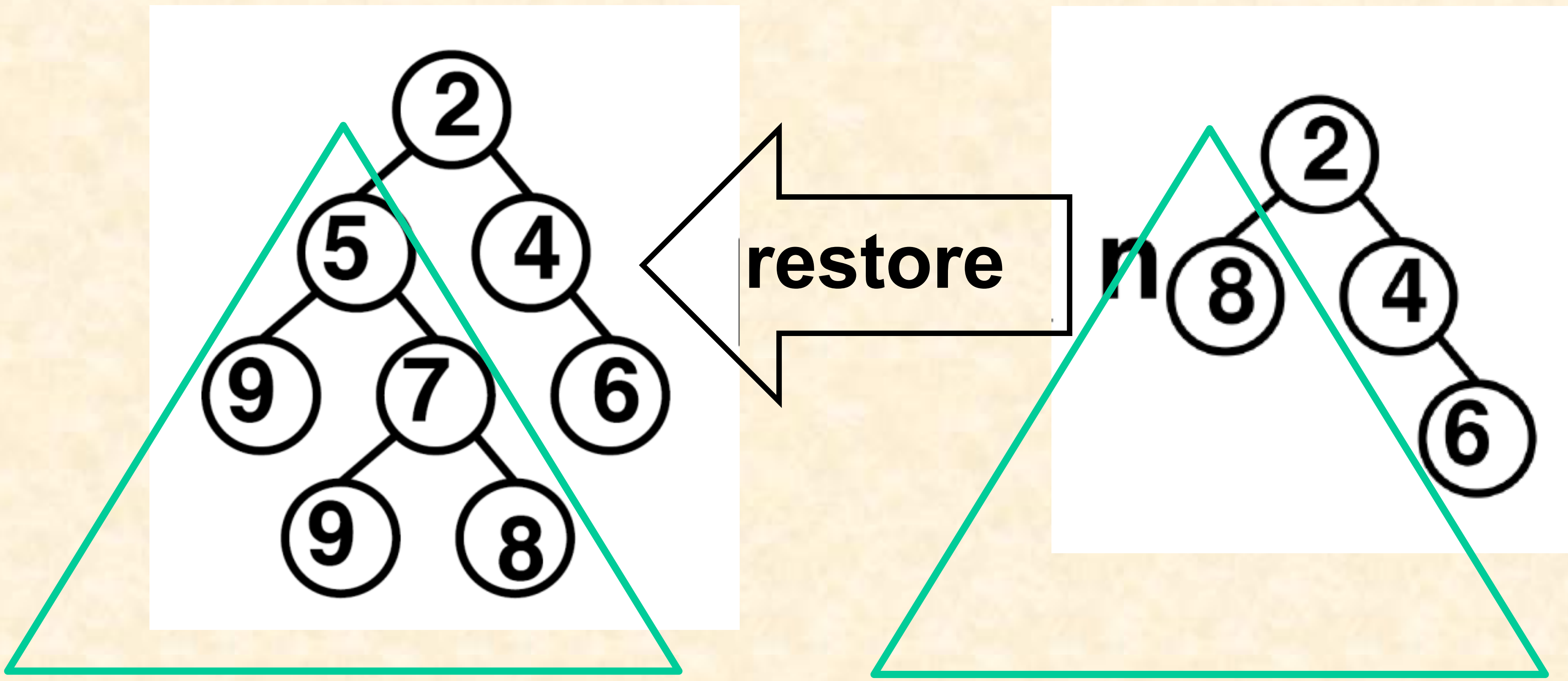
2. Collapse and Restore macros

[#1, SoCS-2015]

2.1 Collapse macro for best-first search



Restore macro



OPEN				
	9	9	8	6

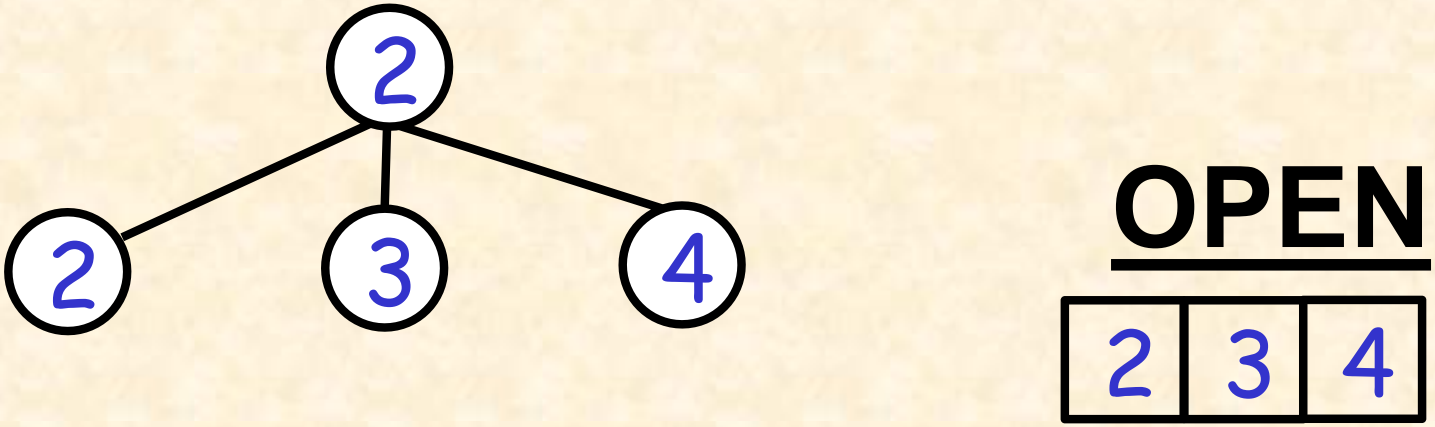
8	6

- Collapse is a **lossy** compression
  1) How do we know a node was collapsed?
  2) How do we restore?

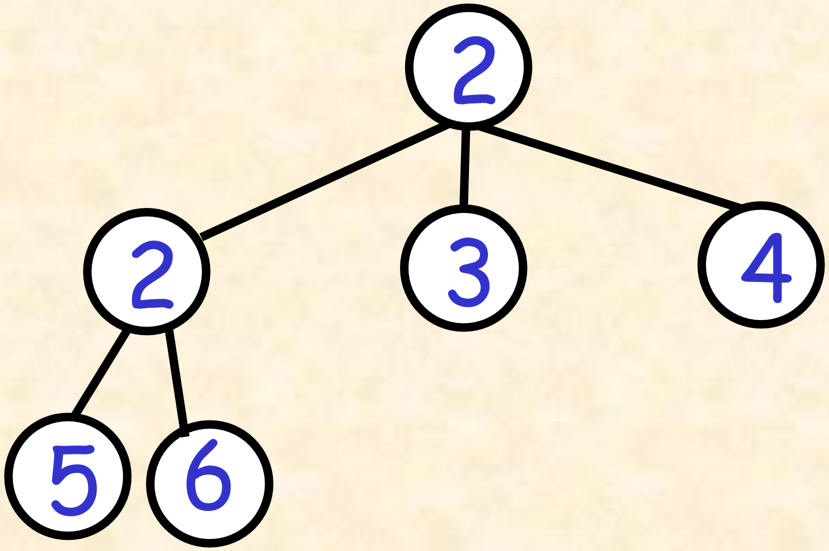
<u>Restore is algorithm dependent</u>

If F(n) > f(n) and the f-value is monotonically increasing just perform a bounded DFS by F(N). [Korf 1993]

2.2 SMA* [#2:Russell 1992]



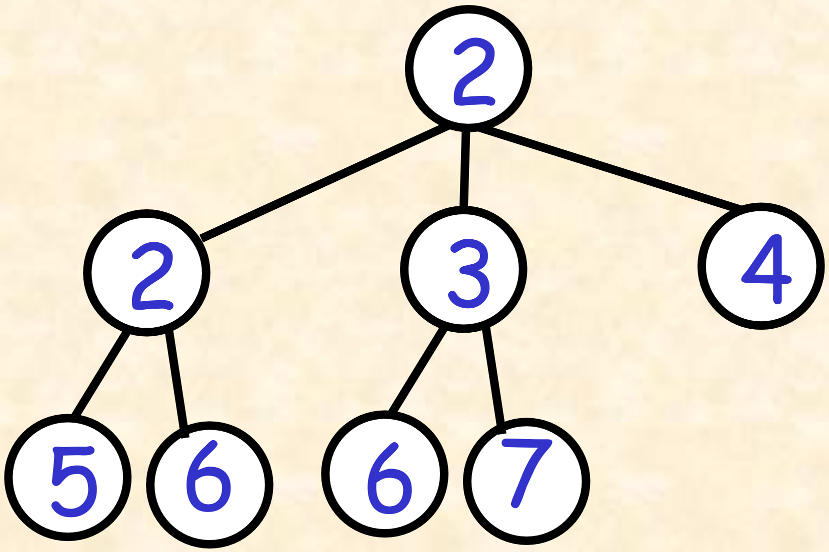
SMA* [Russell 1992]



OPEN

3	4	5	6

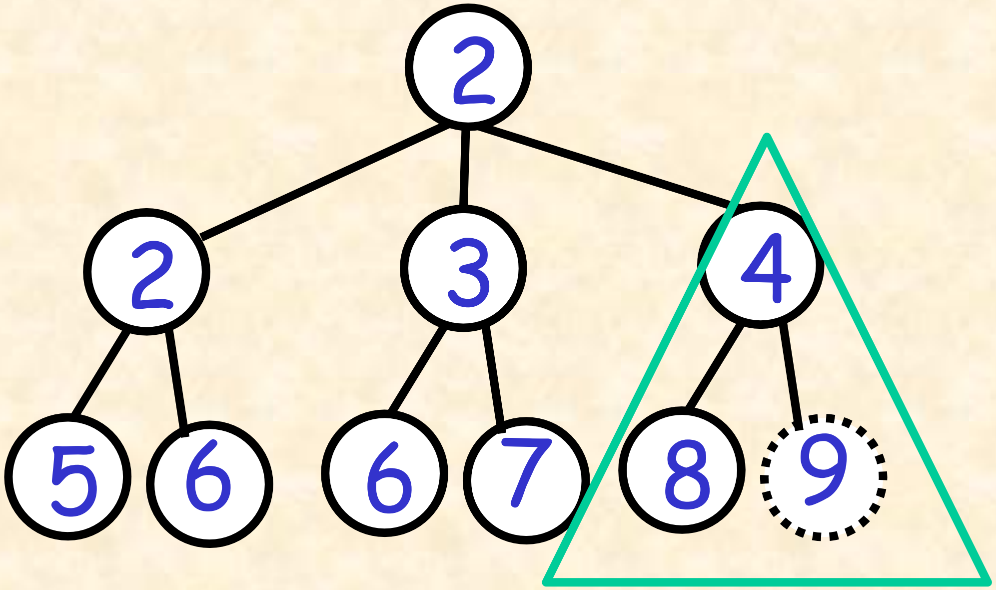
SMA* [Russell 1992]



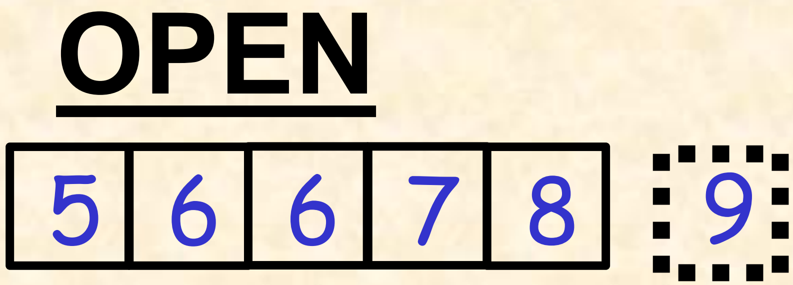
OPEN

4	5	6	6	7

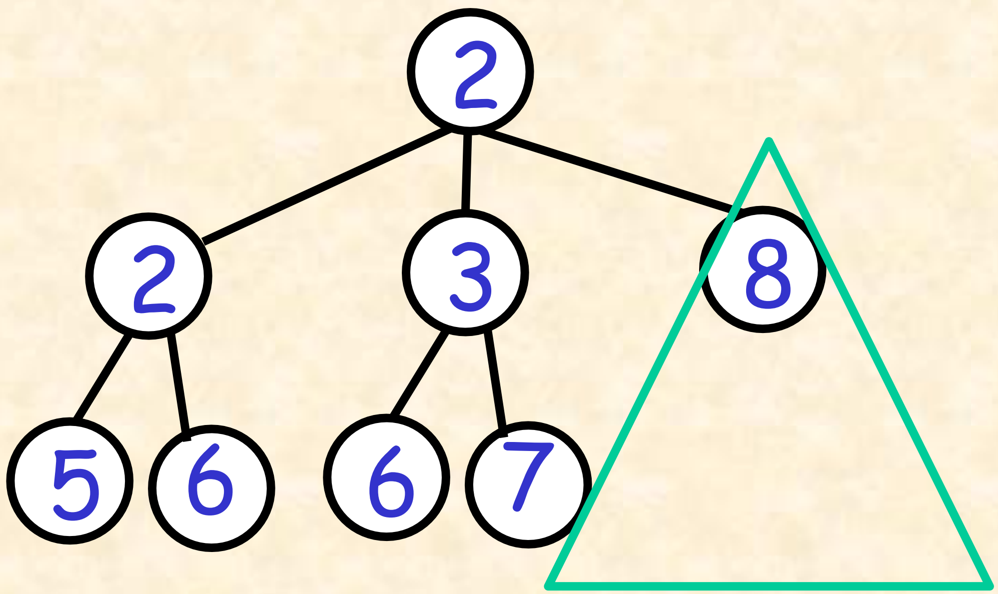
SMA* [Russell 1992]



collapse



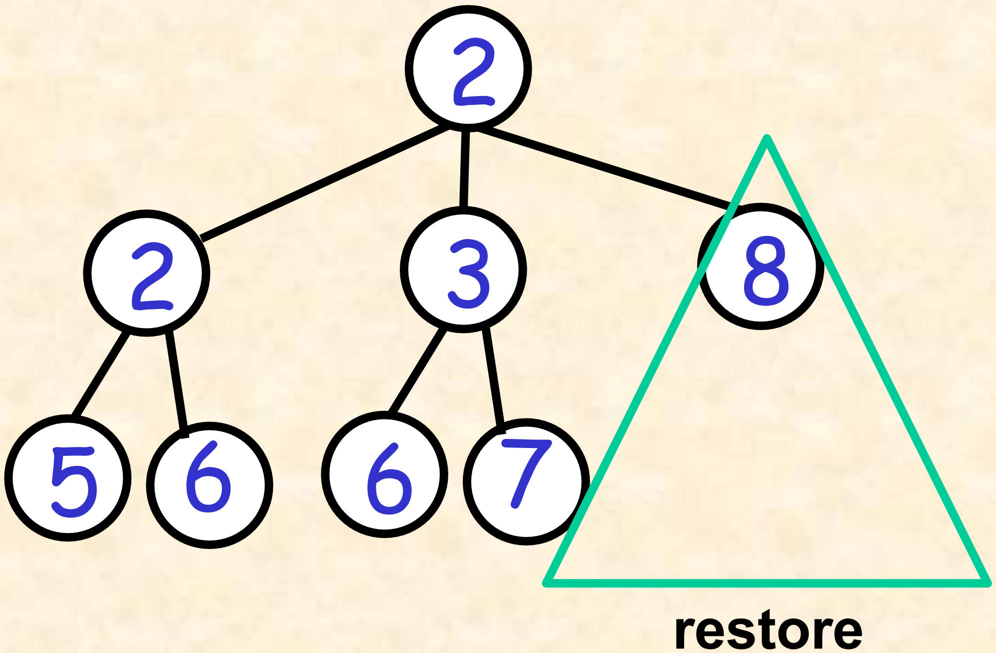
SMA* [Russell 1992]



OPEN

5	6	6	7	8

SMA* [Russell 1992]



OPEN

5	6	6	7	8

SMA* uses a variant of pathmax for its restore macro

2.3. ILBFS [#1 SoCS-2015]

Iterative linear best-first search

Iterative variant of RBFS [Korf, AIJ 1993]

Algorithm 1: High-level ILBFS

```
Input: Root R
Insert R into OPEN and TREE
oldbest=NULL
while OPEN not empty do
    best=extract_min(OPEN)
    if goal(best) then
        exit
```

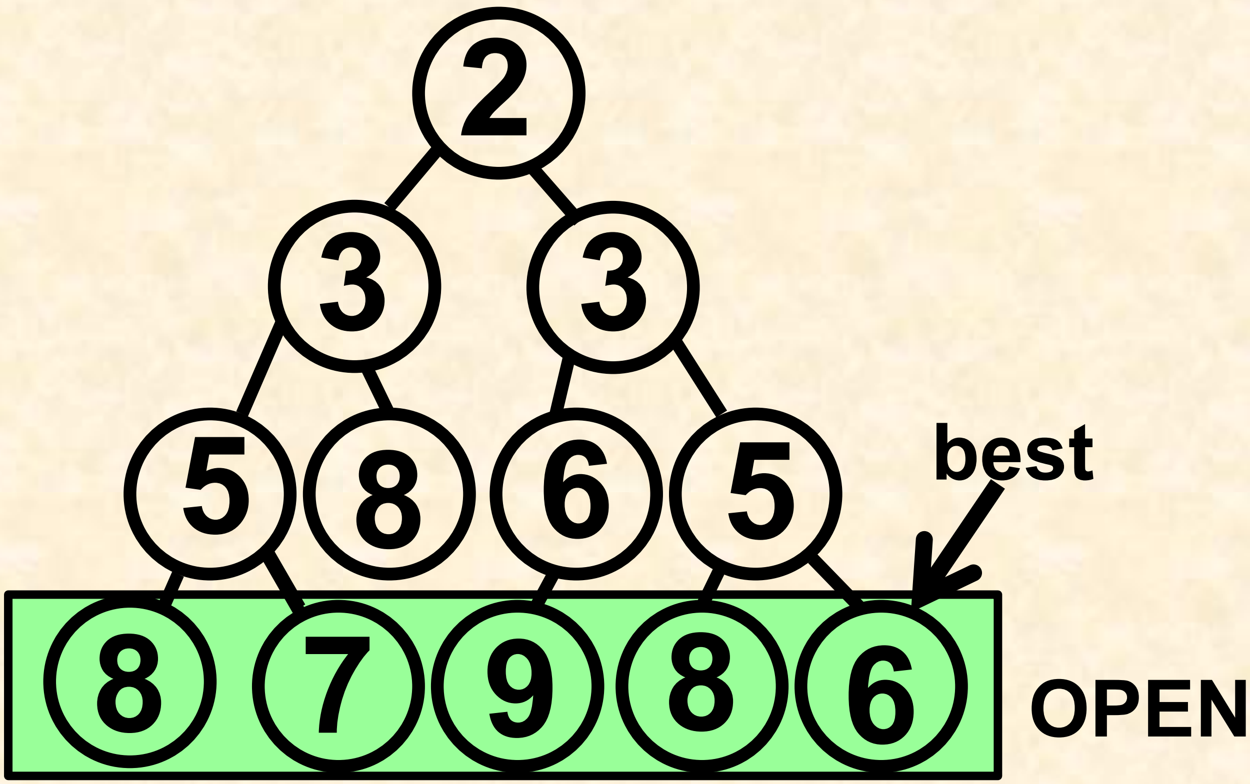
Collapse

Restore

```
    foreach child C of best do
        Insert C to OPEN and TREE
    oldbest ← best
```

- Uses the regular BFS expansion cycle
- Heavily uses the collapse and restore macros

ILBFS

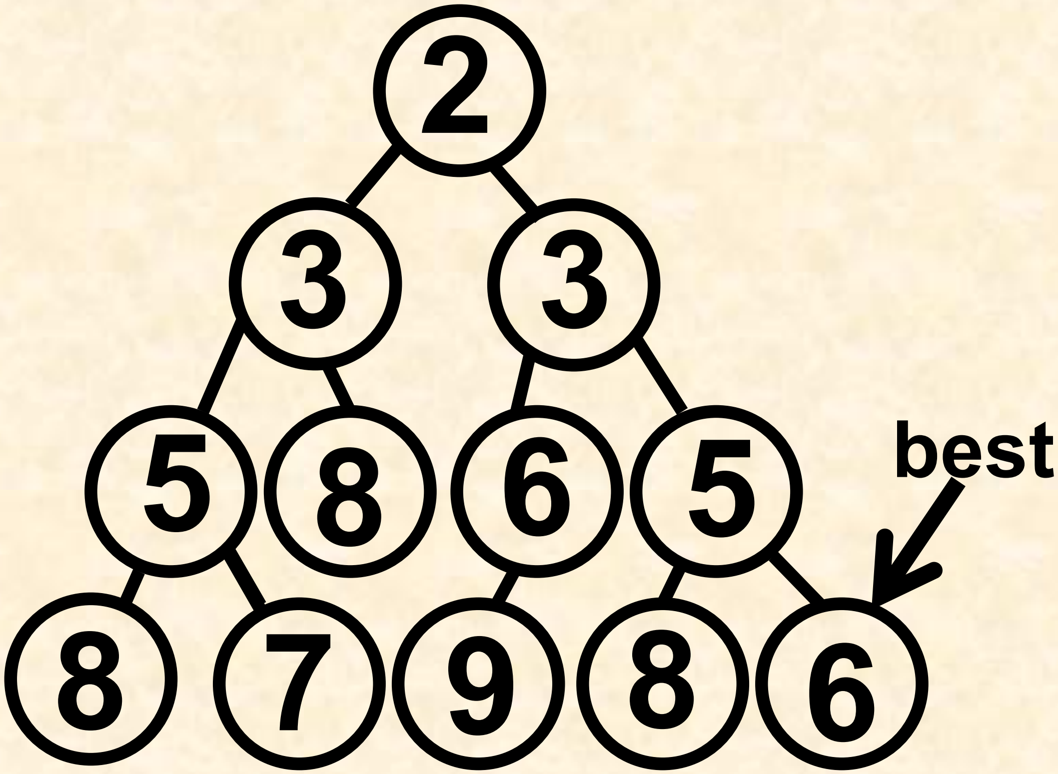


Classic BFS

Principal branch invariant

Store only the branch of the best node and its siblings

ILBFS

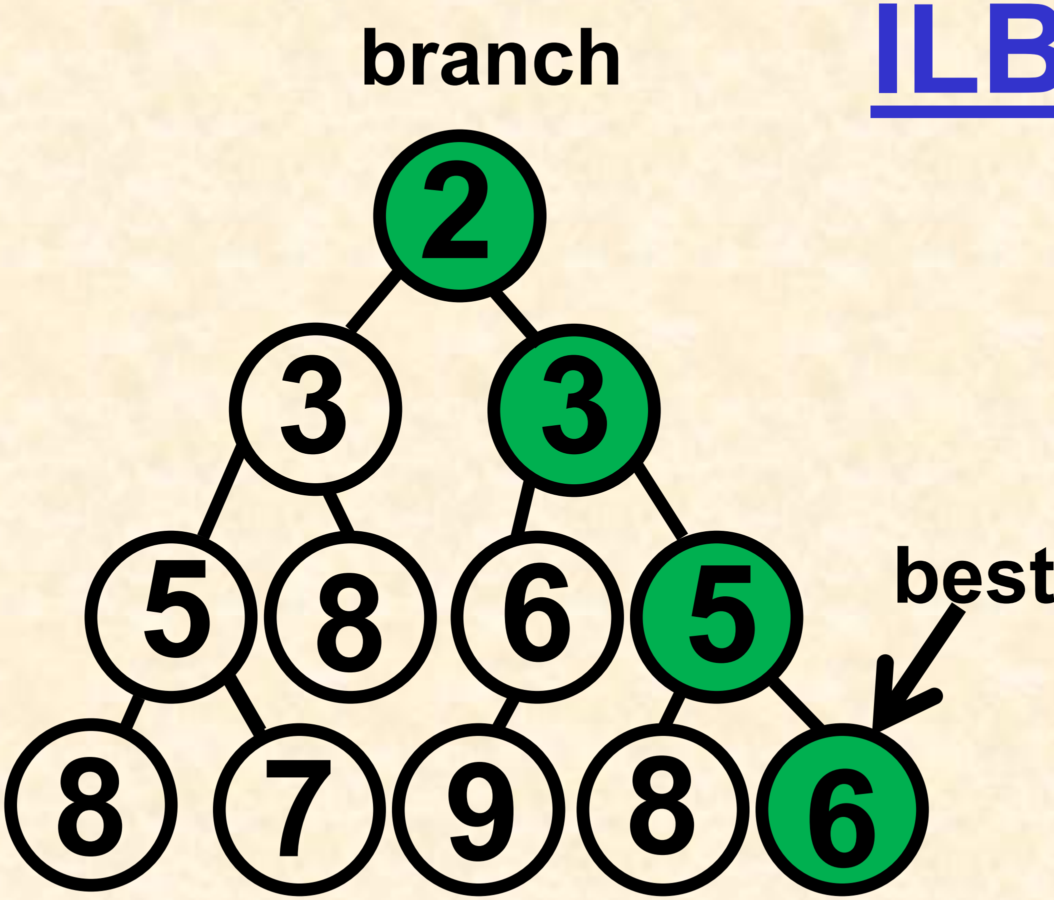


Classic BFS

Principal branch invariant

Store only the branch of the best node and its siblings

ILBFS

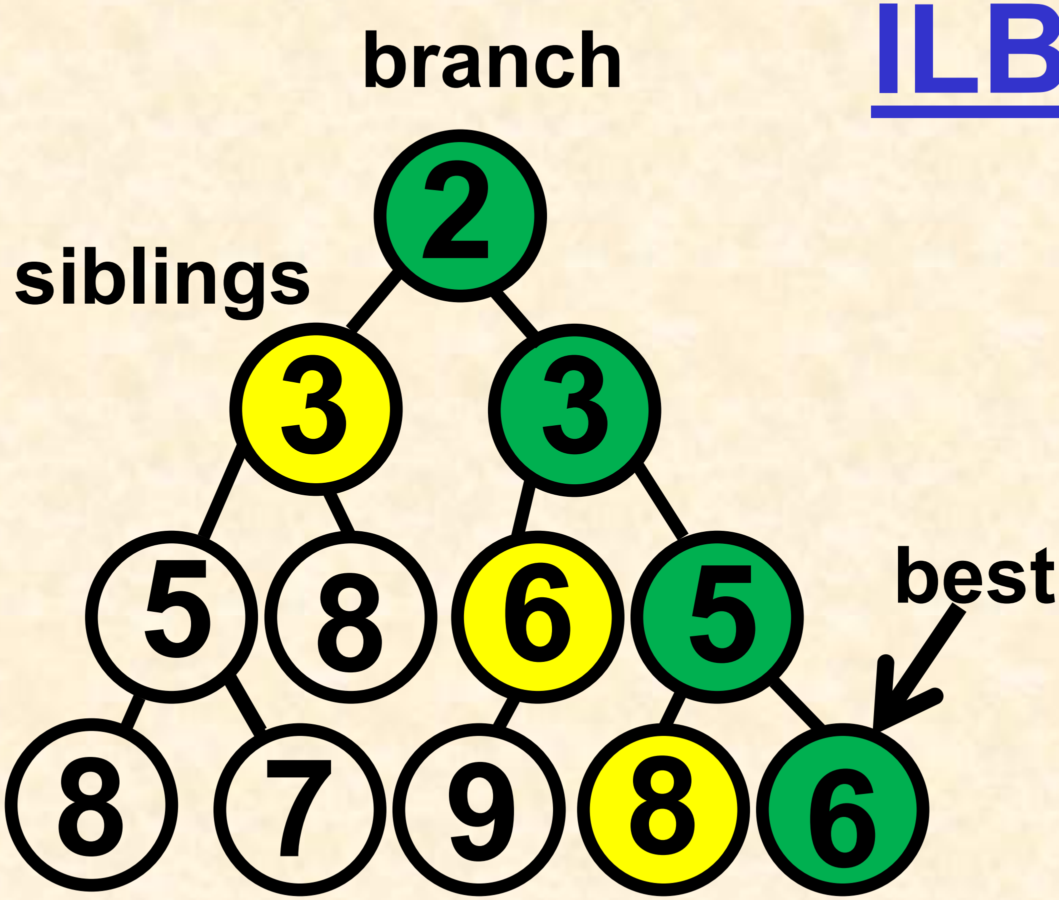


Classic BFS

Principal branch invariant

Store only the branch of the best node and its siblings

ILBFS

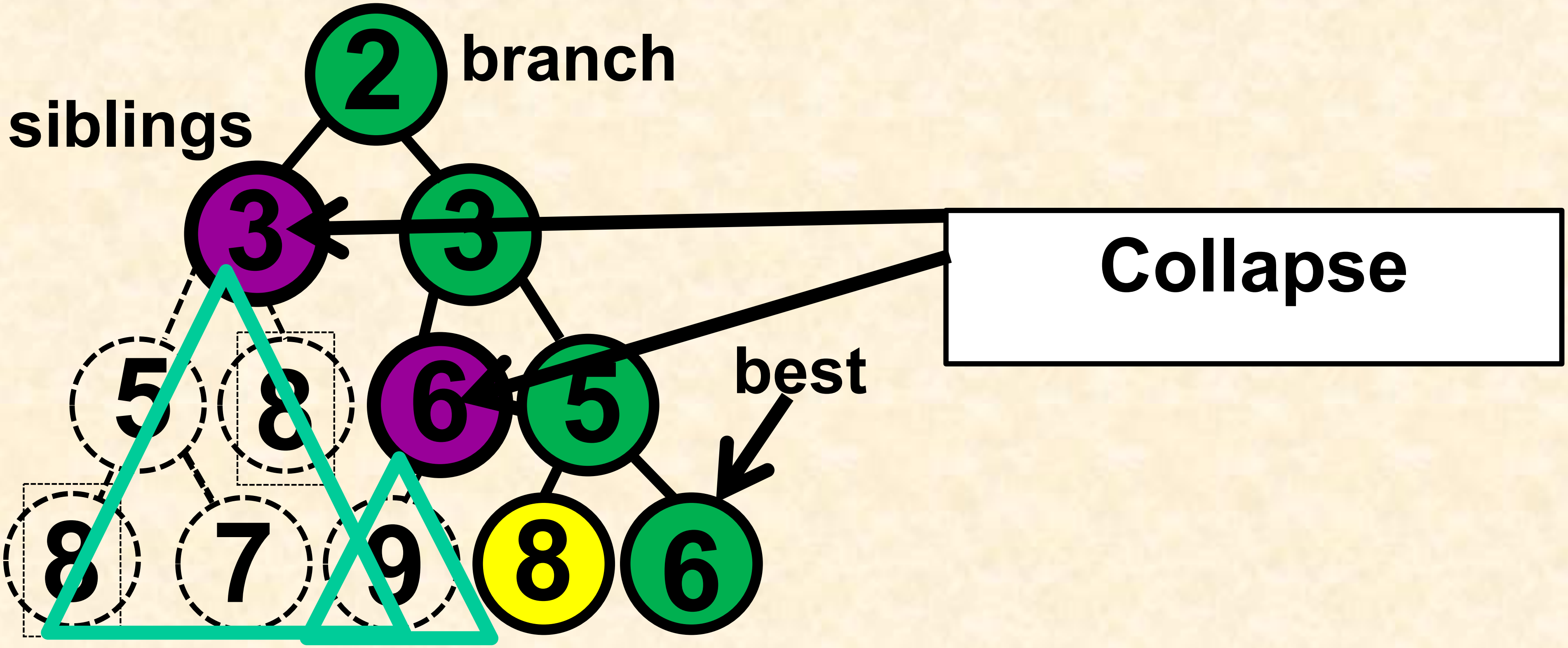


Classic BFS

Principal branch invariant

Store only the branch of the best node and its siblings

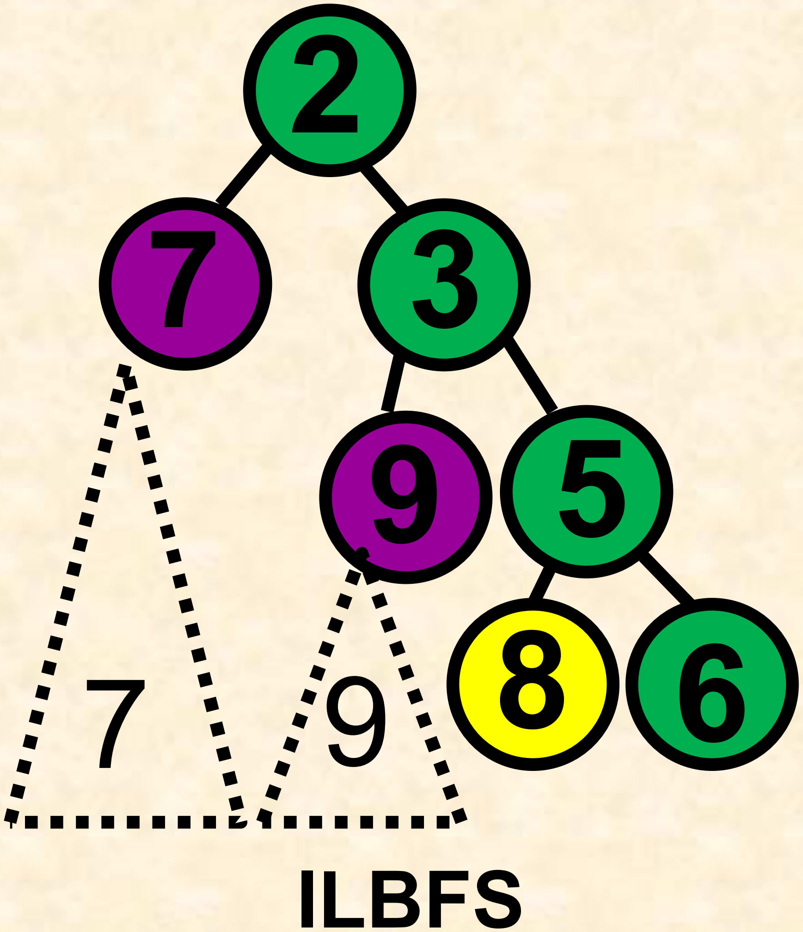
ILBFS



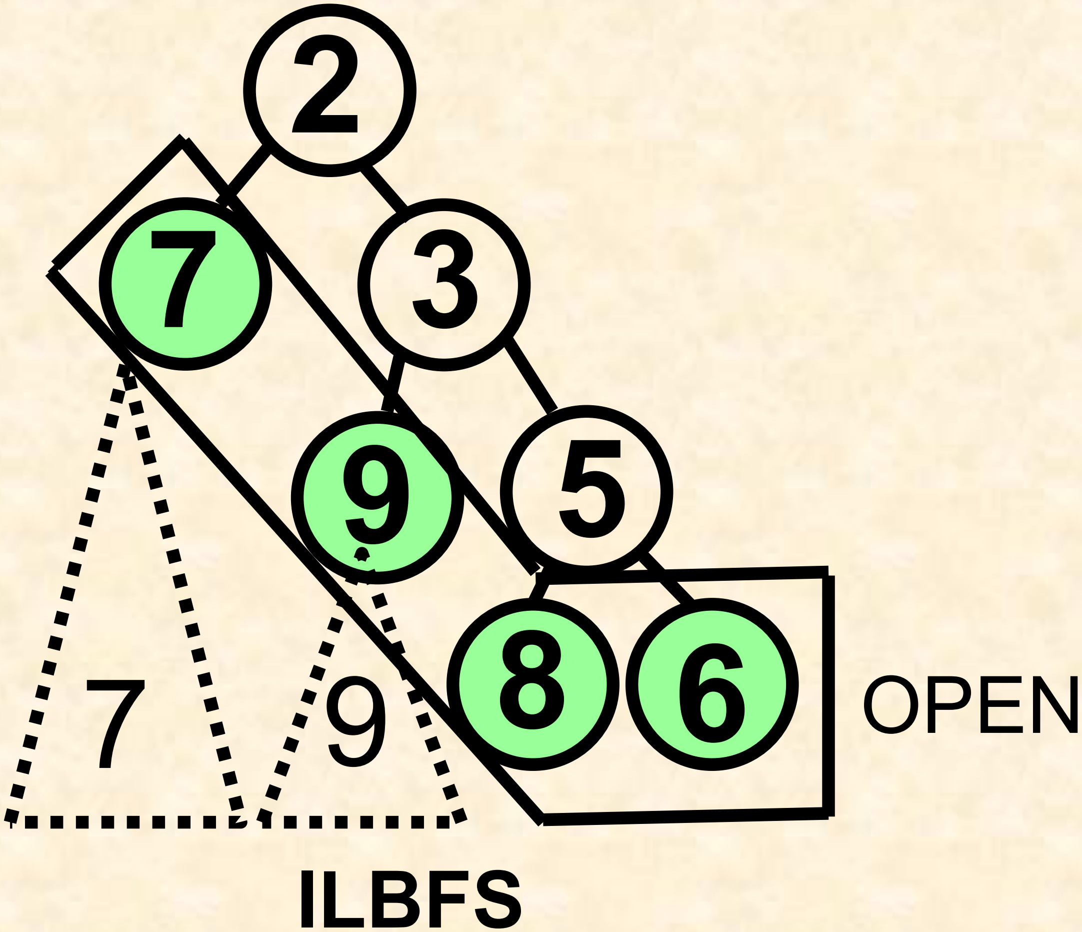
Principal branch invariant

Store only the branch of the best node and its siblings

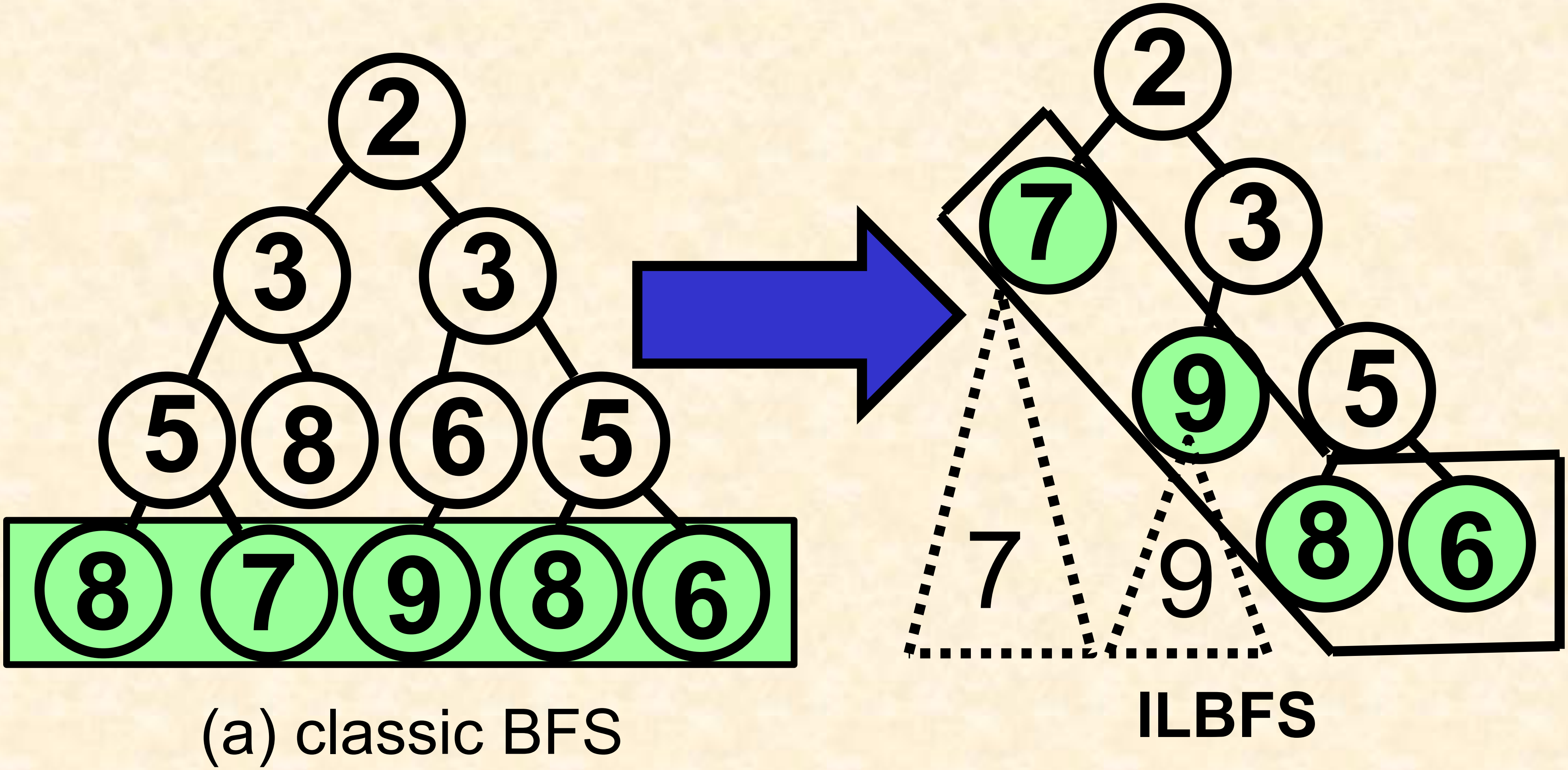
ILBFS



ILBFS



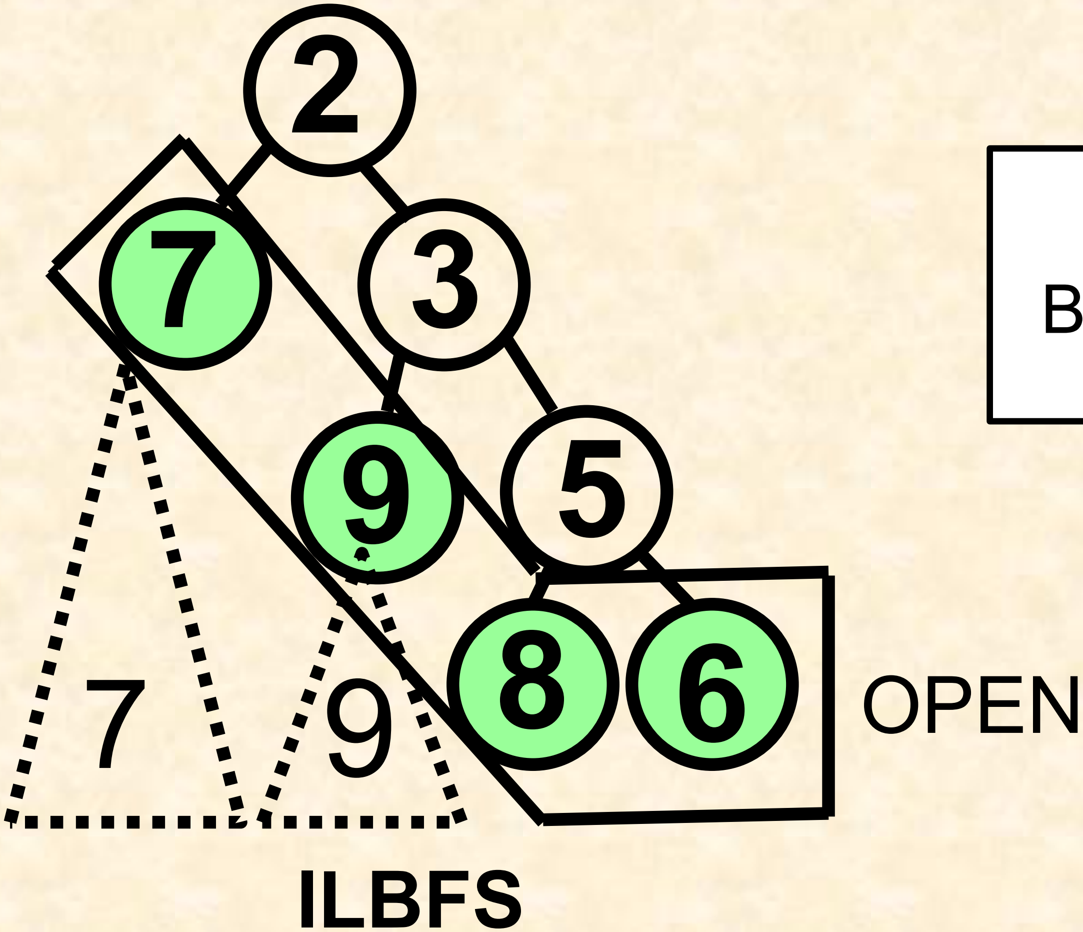
ILBFS



Principal branch invariant

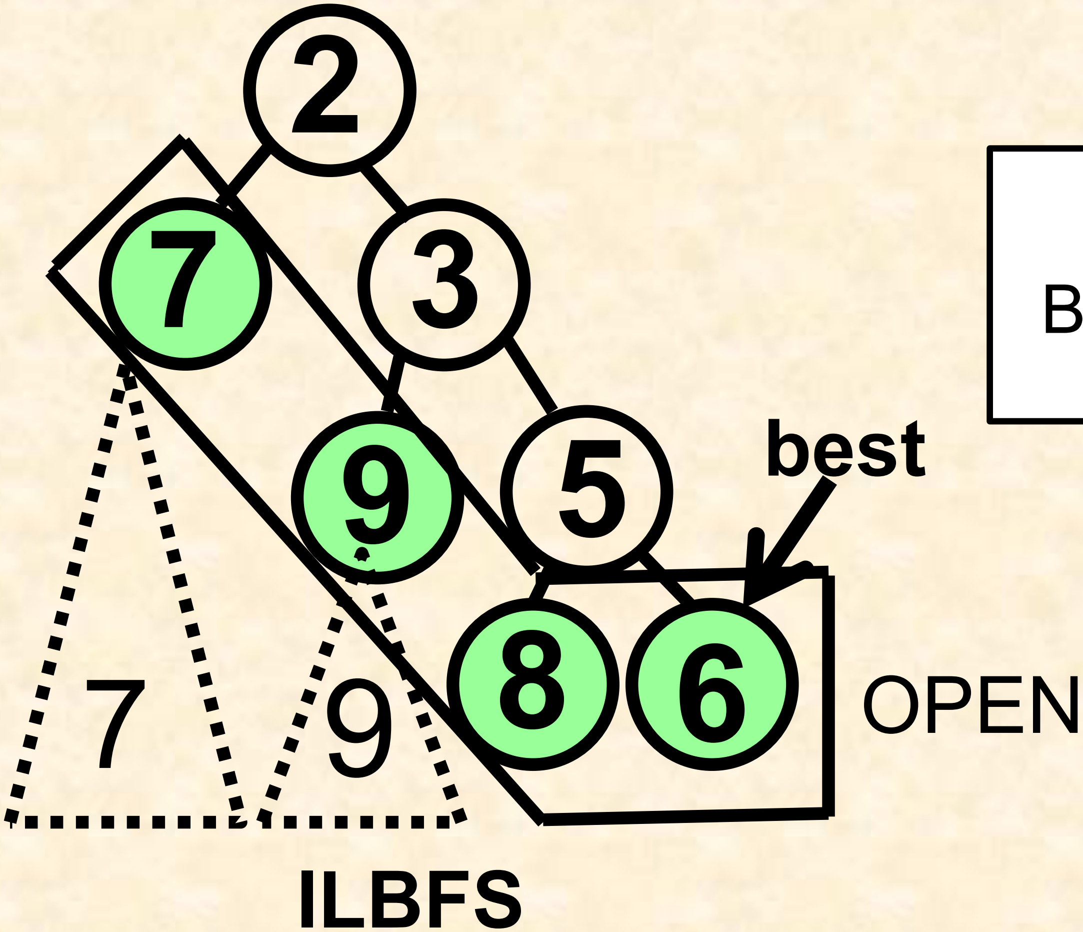
- Initially valid for the root
- Two cases for the expansion cycle.

ILBFS



Case 1:
Best is a child of oldbest

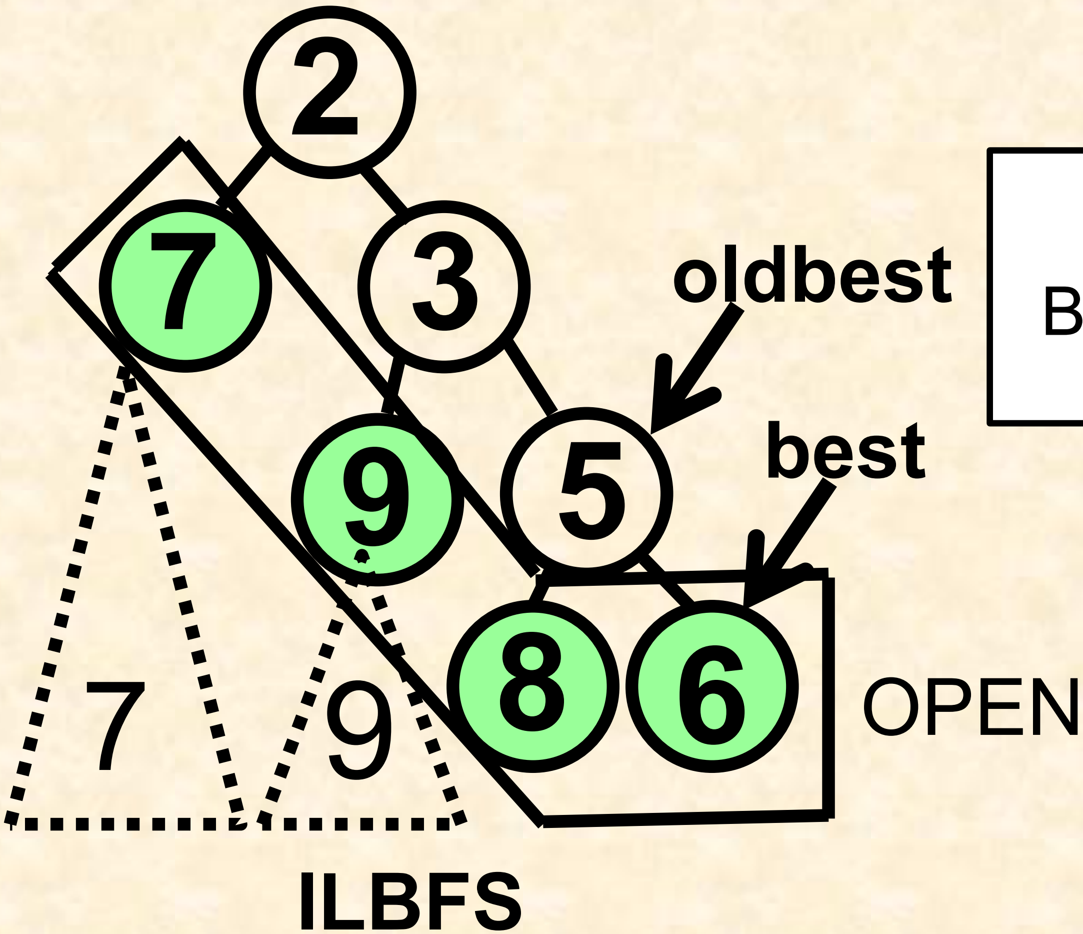
ILBFS



Case 1:

Best is a child of oldbest

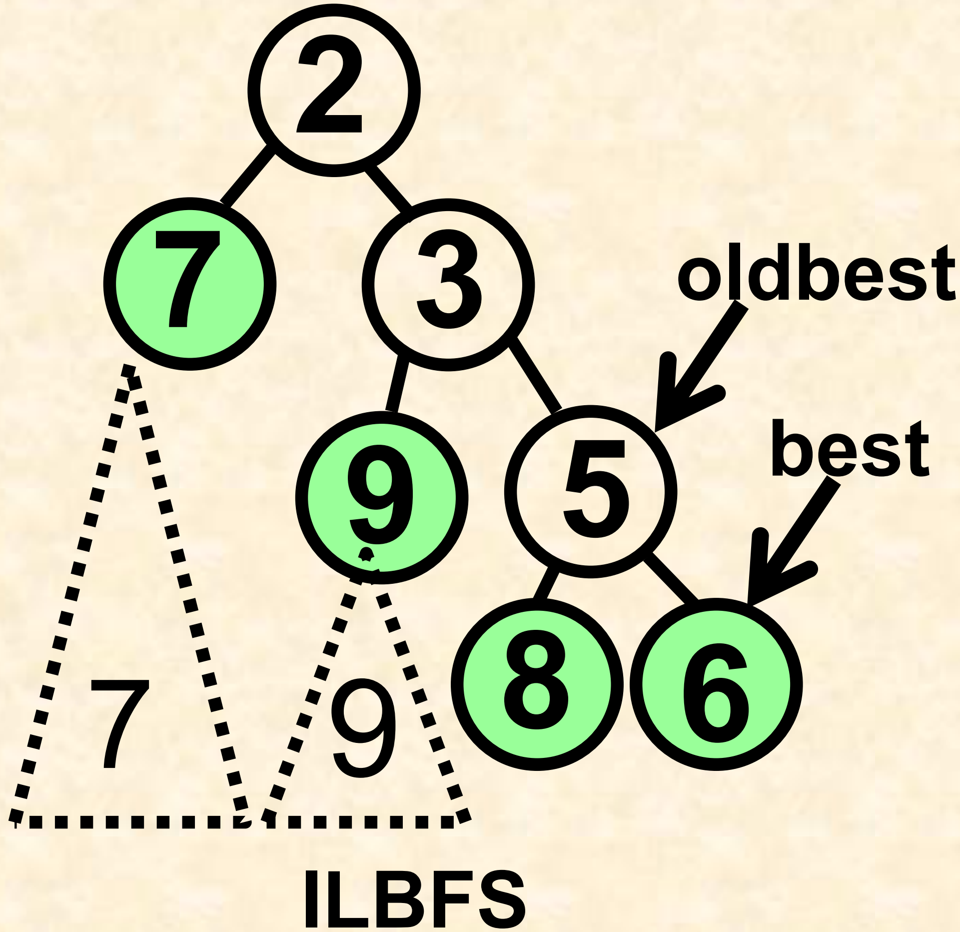
ILBFS



Case 1:

Best is a child of oldbest

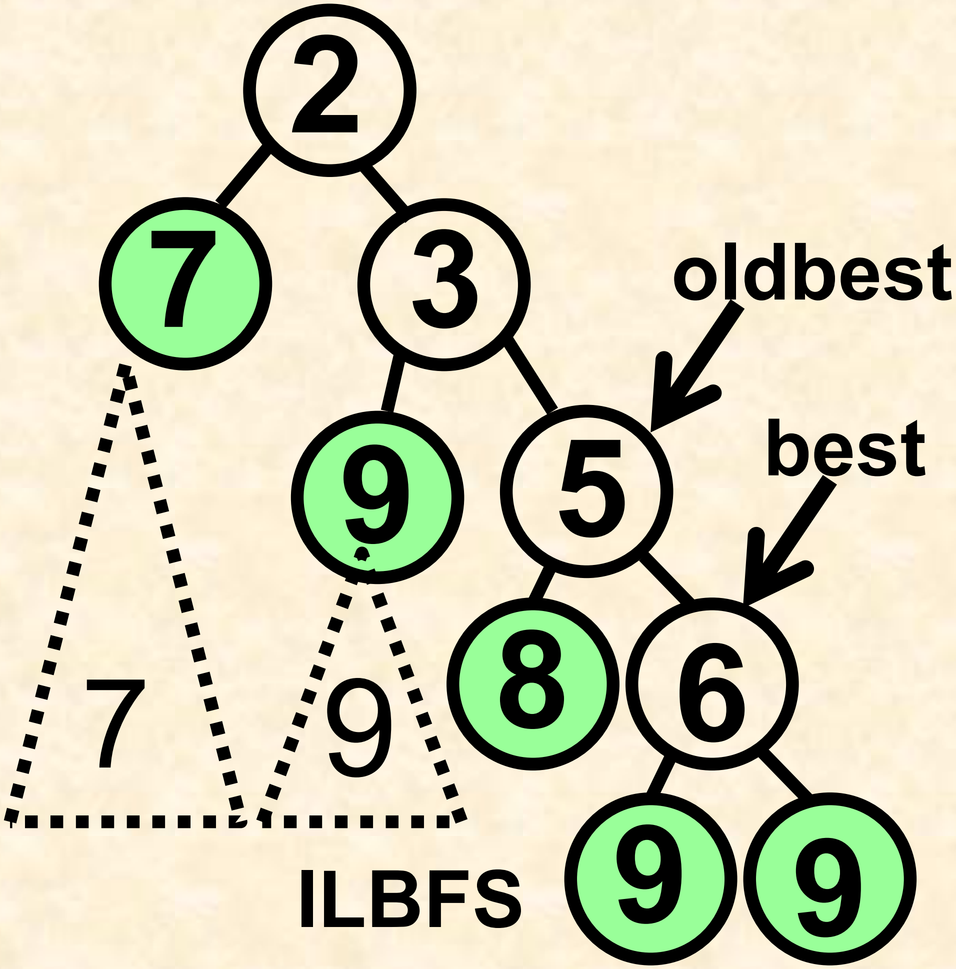
ILBFS



Case 1:

Best is a child of oldbest

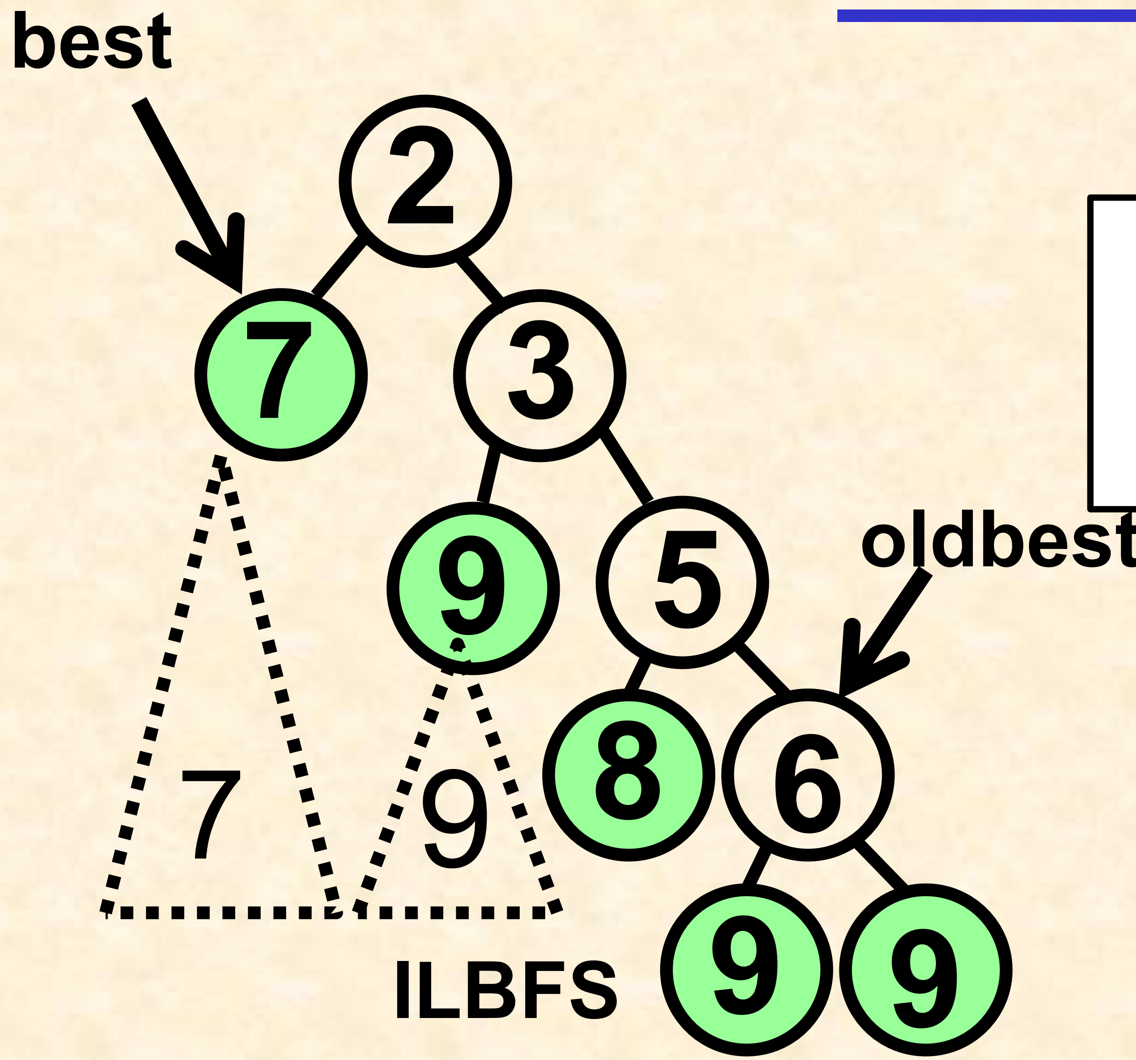
ILBFS



Case 1:

Best is a child of oldbest

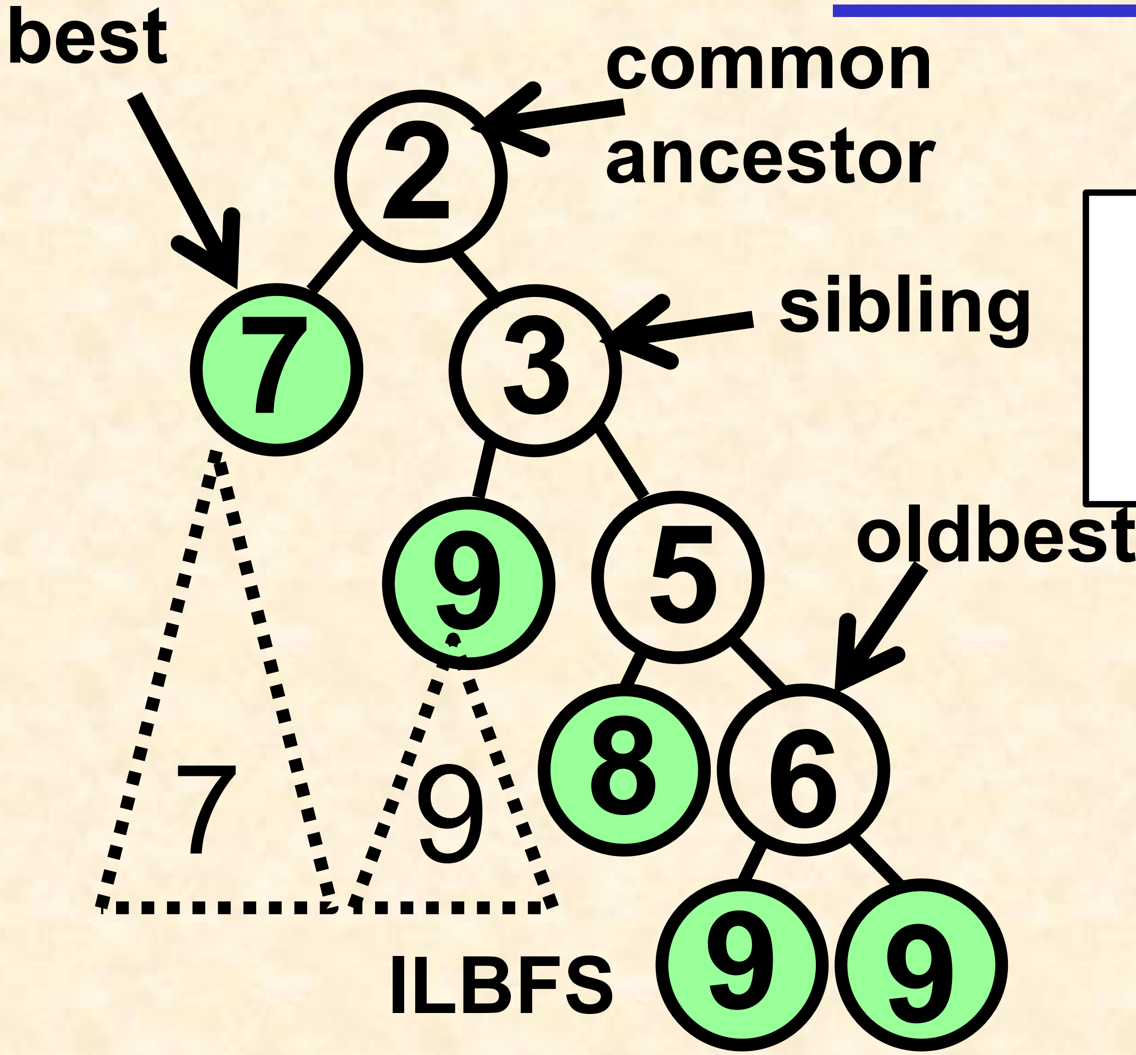
ILBFS



Case 2:

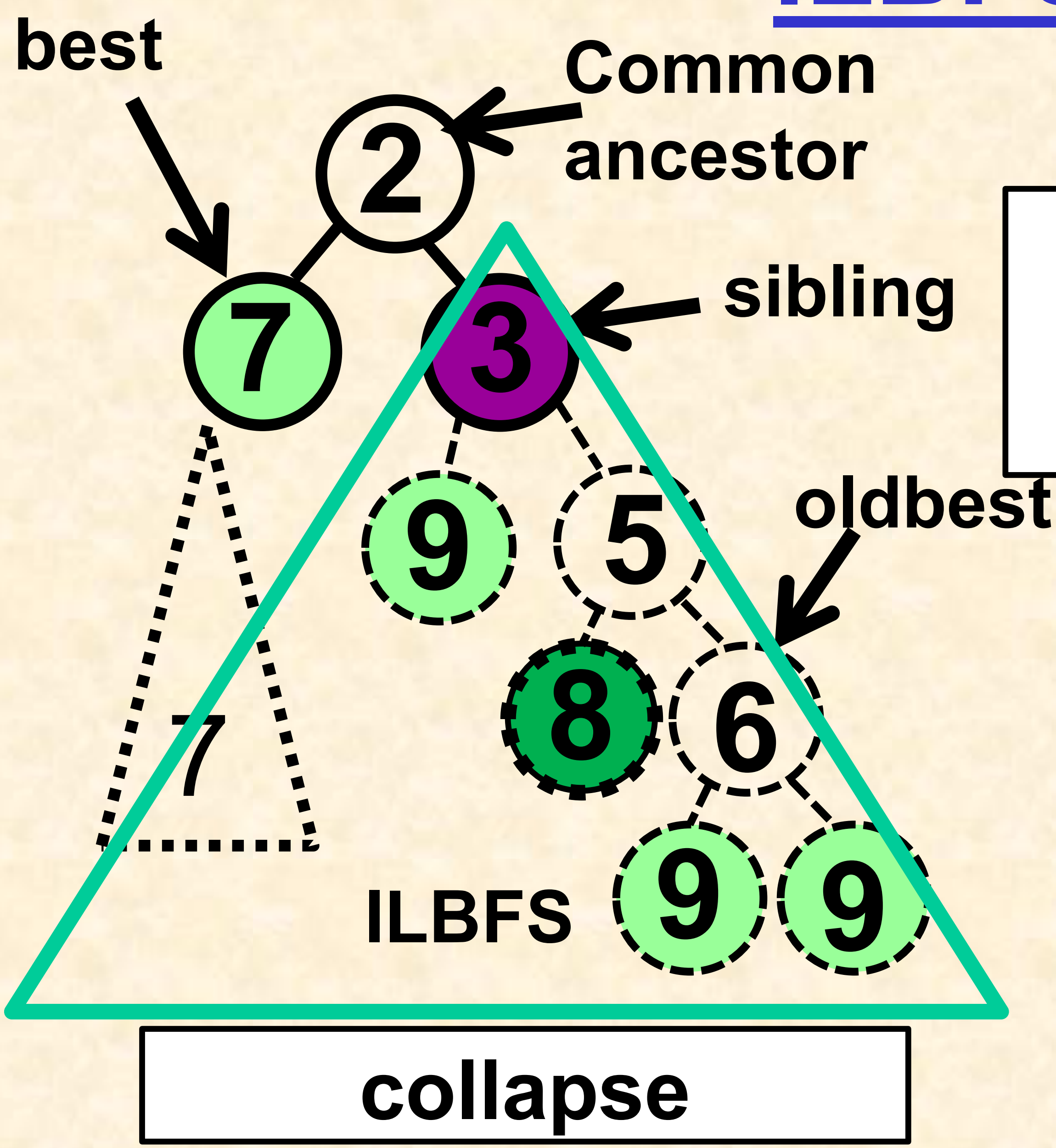
Best is not a child of oldbest

ILBFS



Case 2:
Best is not a
child of oldbest

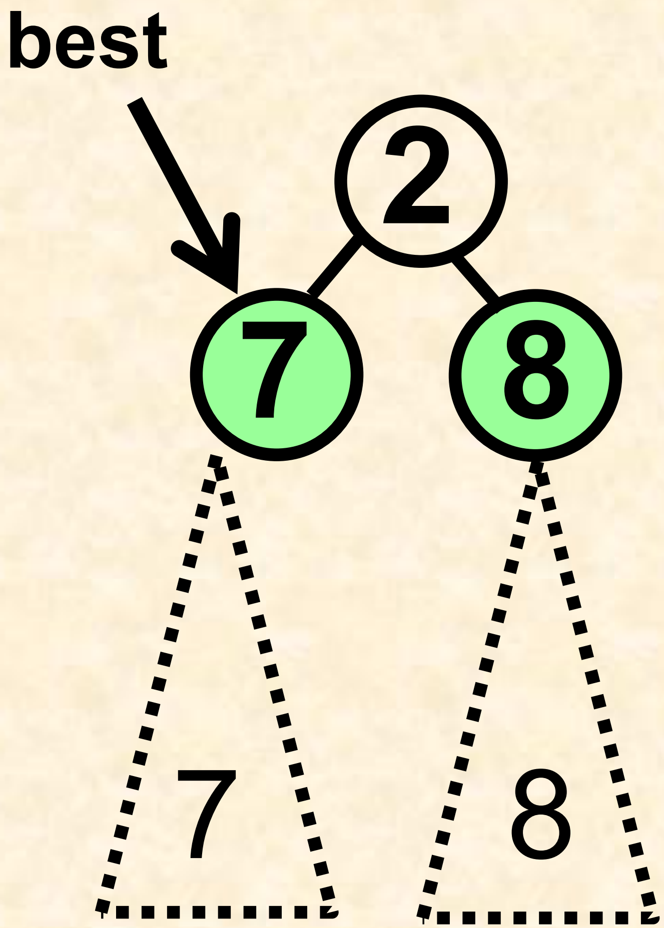
ILBFS



Case 2:

Best is not a child of oldbest

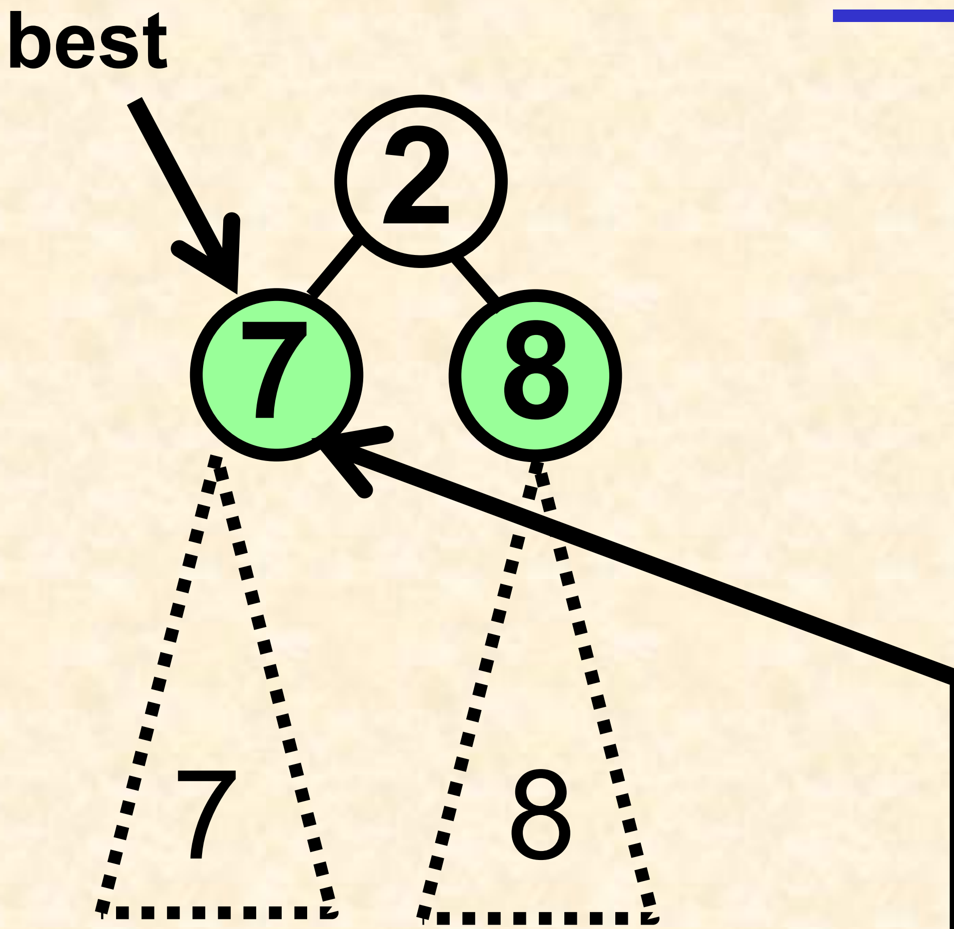
ILBFS



Case 2:

Best is not a child of oldbest

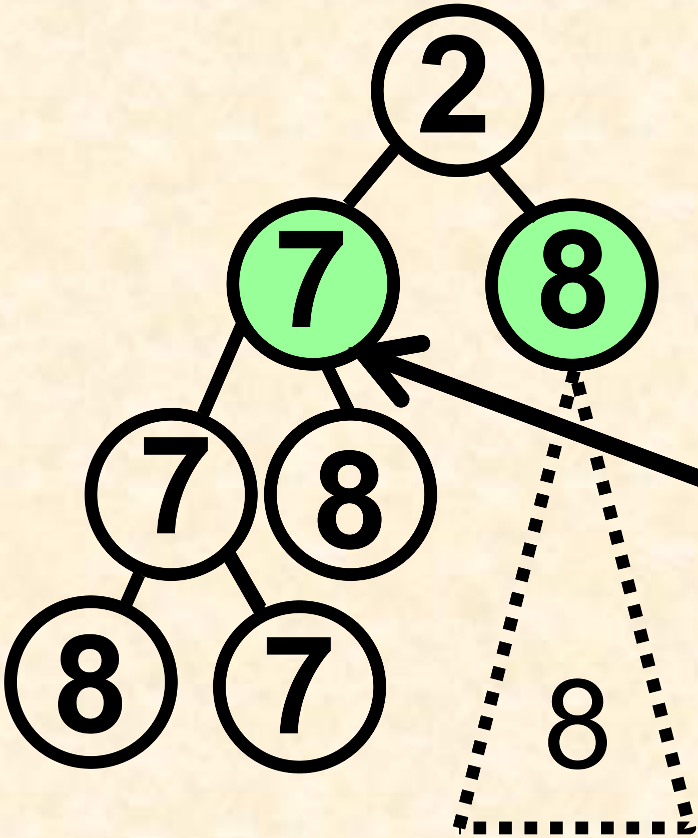
ILBFS



Case 2:
Best is a collapsed node

Restore
f=3, F=7
DFS(7)

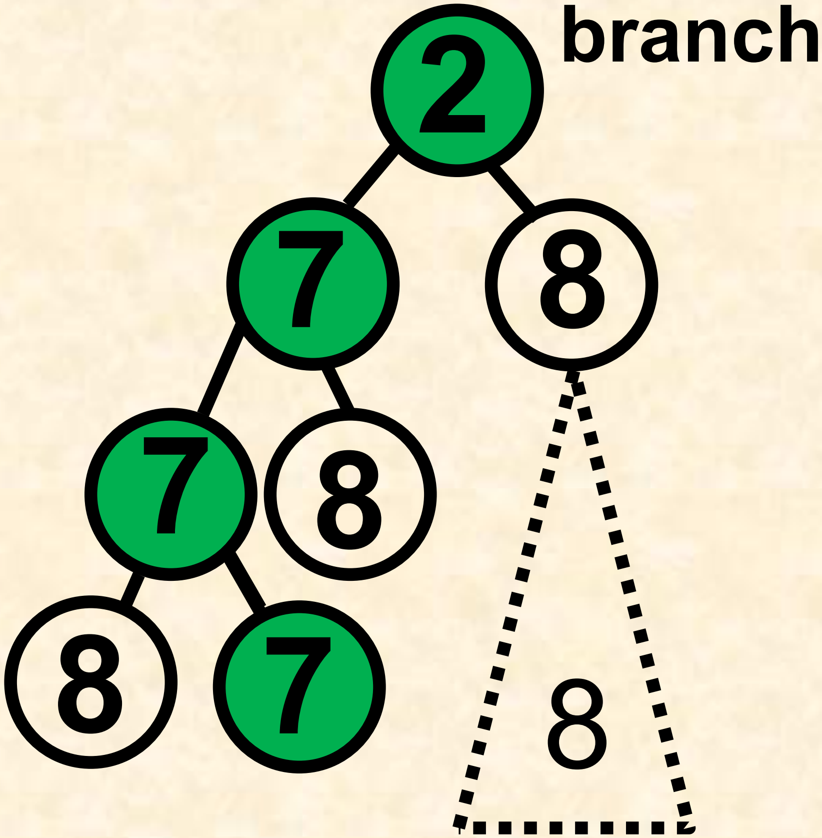
ILBFS



Case 2:
Best is a collapsed node

Restore
f=3, F=7
DFS(7)

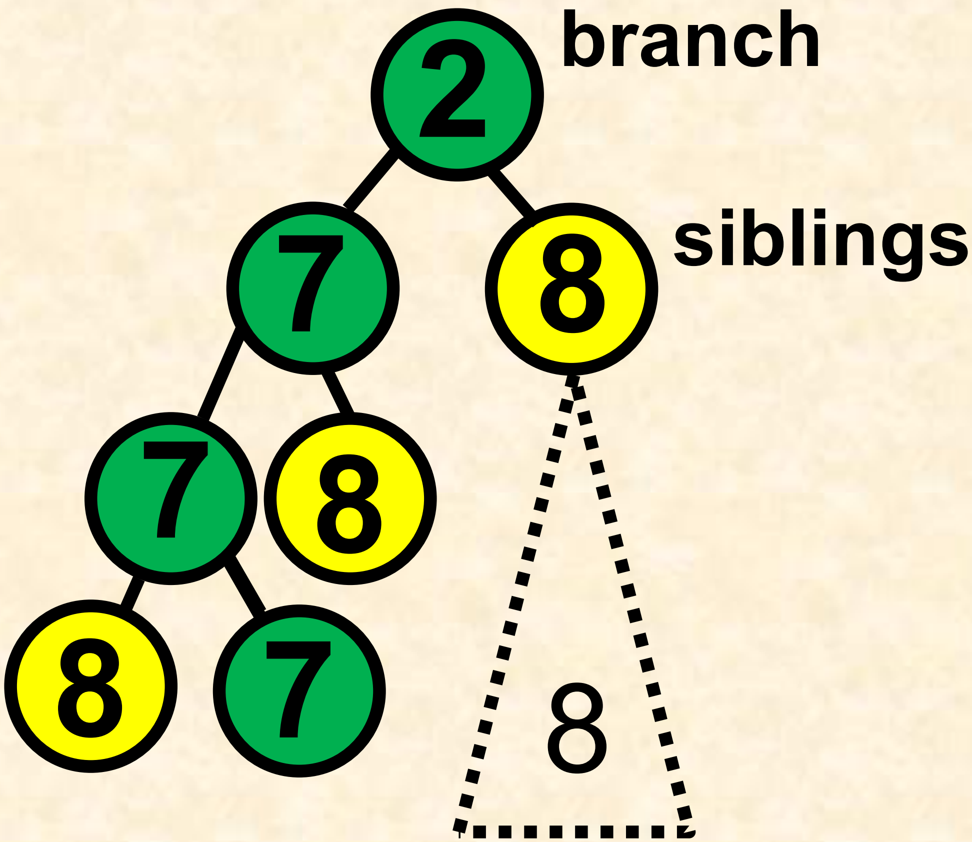
ILBFS



Case 2:

Best is a collapsed node

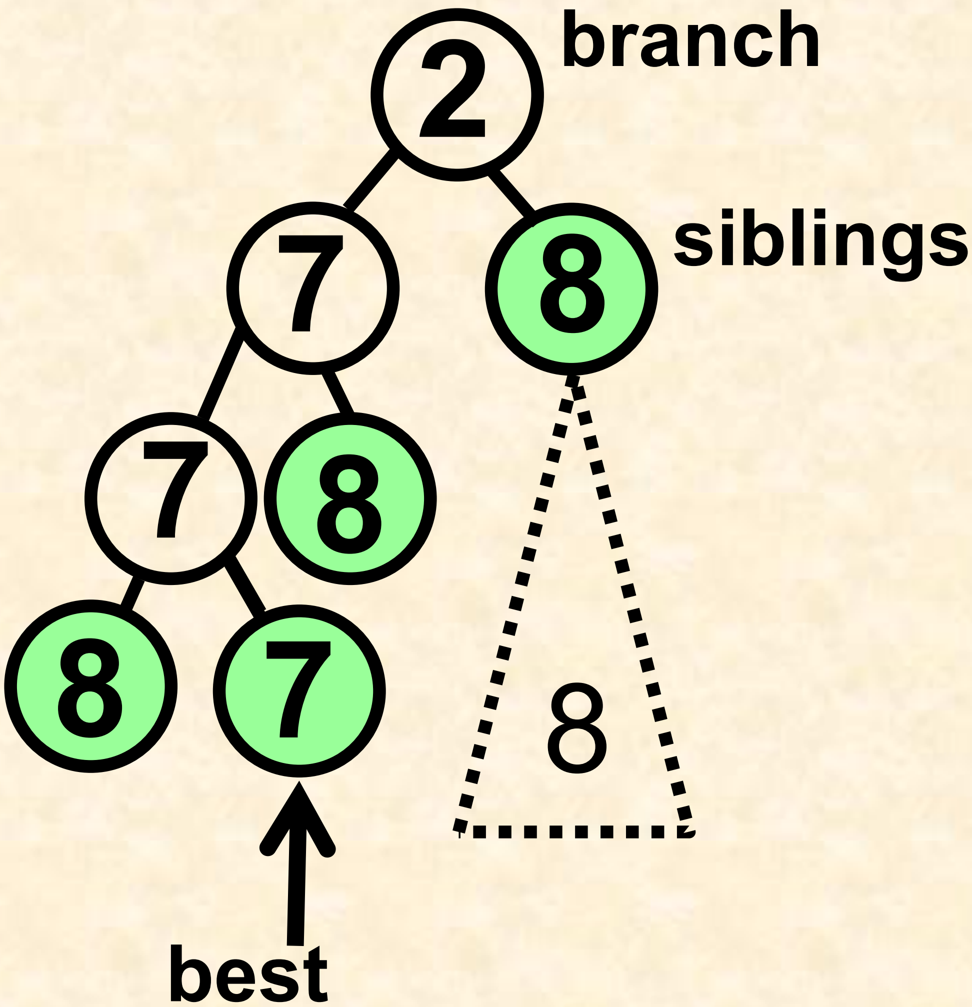
ILBFS



Case 2:

Best is a collapsed node

ILBFS



Case 2:

Best is a collapsed node

Restore in ILBFS

- Restore is the only non-trivial step of ILBFS (and RBFS too)

- Observation: After collapse F(n) > f(n)

- Restore: DFS below n bounded by F(n)

Linear-space best-first search

Iterative variant - ILBFS

Algorithm 1: High-level ILBFS

Input: Root R

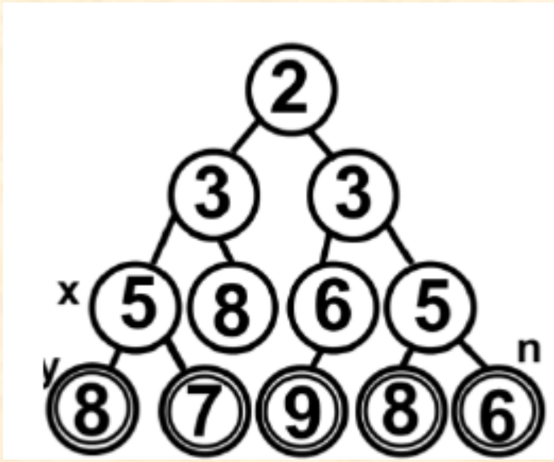
Insert R into OPEN and TREE
oldbest=NULL
while *OPEN not empty* **do**
    best=extract_min(OPEN)
    if *goal(best)* **then**
        exit
    if $oldbest \neq best.parent$ **then**
        $B \leftarrow$ sibling of $oldbest$ that is ancestor of $best$
        collapse(B)
    if $best.C$=$True$ **then**
        $best \leftarrow$ restore($best$)
    foreach *child* C *of best* **do**
        Insert C to OPEN and TREE
    $oldbest \leftarrow best$

Recursive variant - RBFS

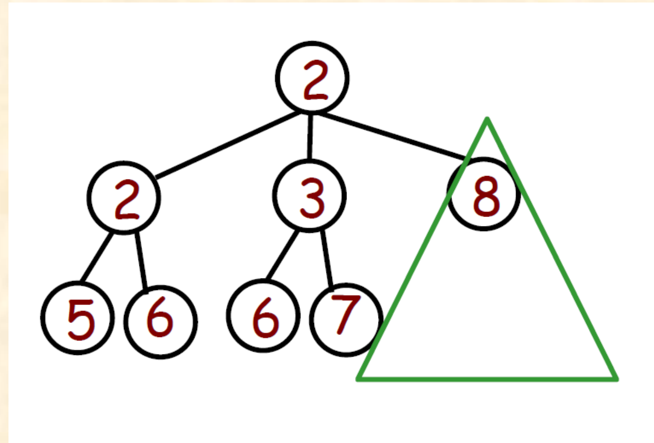
RBFS(n, B)
1. if n is a goal
2. $solution \leftarrow n$; exit()
3. $C \leftarrow expand(n)$
4. if C is empty, return ∞
5. for each child n_i in C
6.   if $f(n) < F(n)$ then $F(n_i) \leftarrow max(F(n), f(n_i))$
7.   else $F(n_i) \leftarrow f(n_i)$
8. $(n_1, n_2) \leftarrow best_{\mathrm{F}}(C)$
9. while $(F(n_1) \leq B$ and $F(n_1) < \infty)$
10. $F(n_1) \leftarrow$ RBFS$(n_1, min(B, F(n_2)))$
11. $(n_1, n_2) \leftarrow best_{\mathrm{F}}(C)$
12. return $F(n_1)$

ILBFS – an iterative variant of RBFS [Korf 1993]

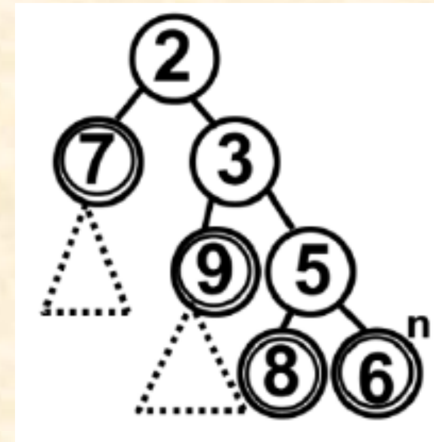
A*



SMA*



RBFS/ILBFS



memory	b^d	*M*	*O(d)*
collapse	never	lazily	eagerly

continuum