Multiway mergesort

Röðun byggð á samröðun margra raðaðra runa

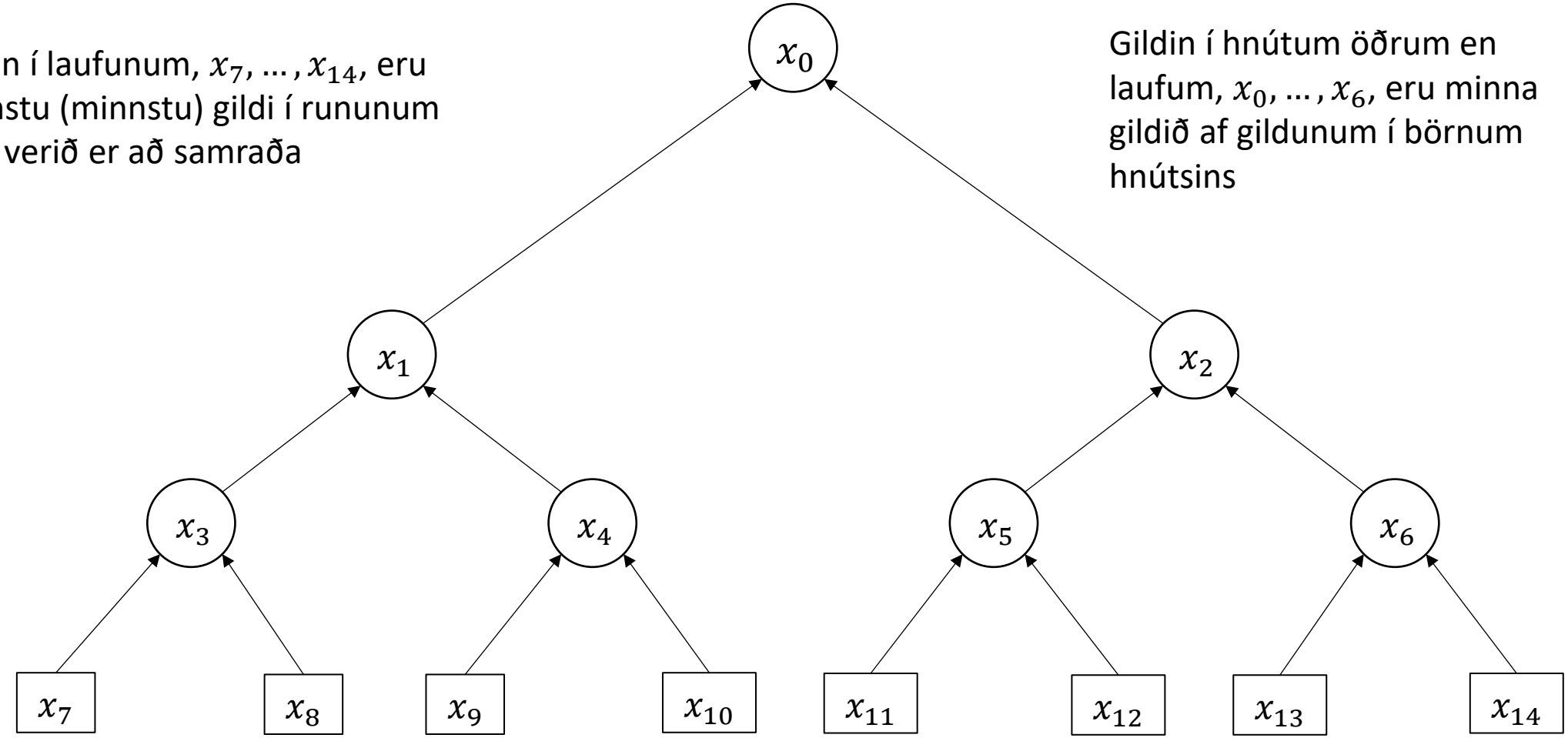
Grunnhugmynd multiway mergesort

- Skiptum poka þeirra gilda sem raða skal í k jafnstóra poka $S_1, \ldots, S_k$
- Röðum hverjum poka S_i og fáum runu R_i
- Samröðum rununum og fáum raðaða heildarrunu

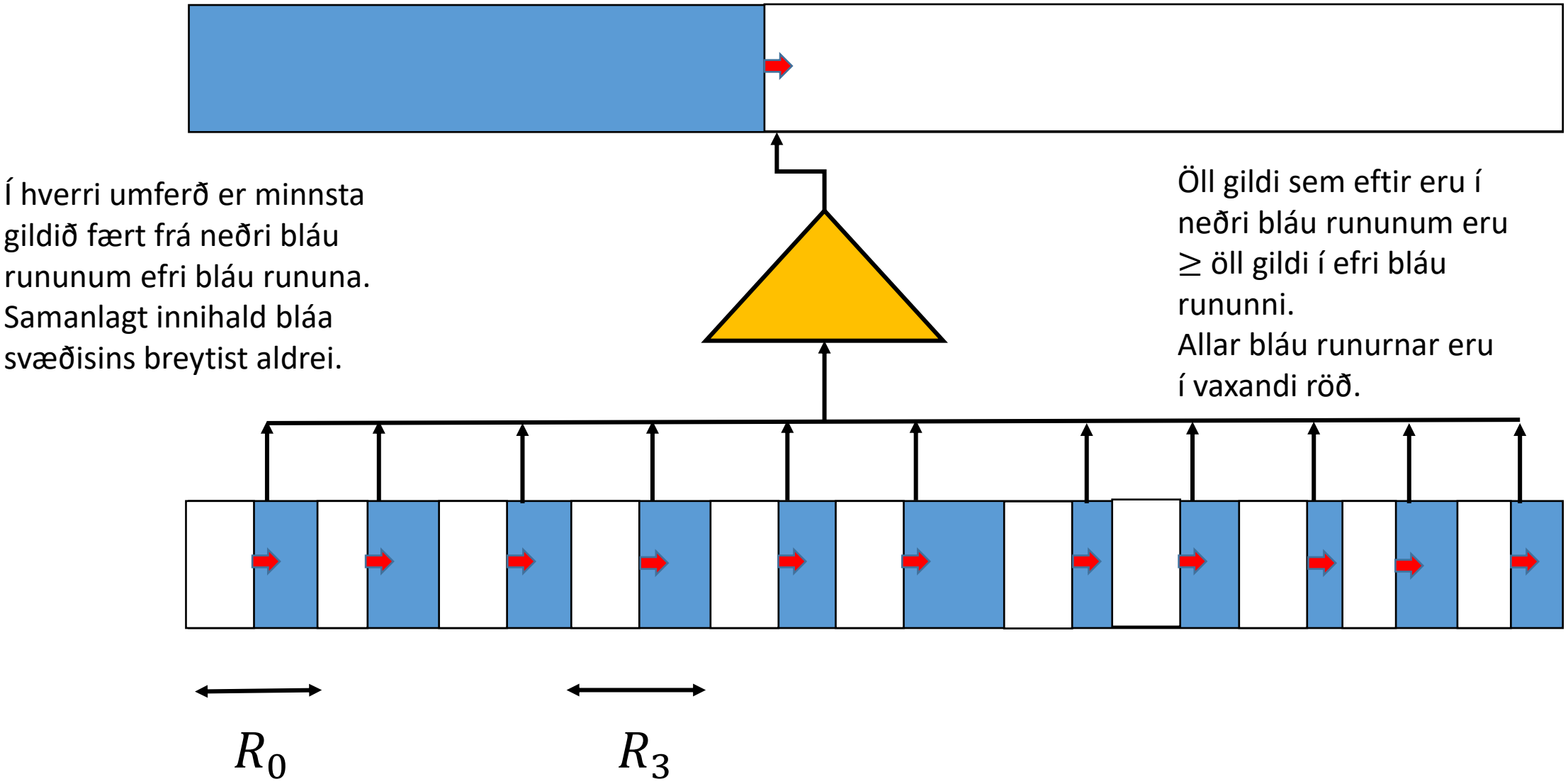
Samröðunartré (útsláttarkeppni gilda)

Gildin í laufunum, $x_7, \ldots, x_{14}$, eru fremstu (minnstu) gildi í rununum sem verið er að samraða

Gildin í hnútum öðrum en laufum, $x_0, \ldots, x_6$, eru minna gildið af gildunum í börnum hnútsins



Fjölrunu samröðunarlykkja



Kostir og gallar multiway mergesort

Kostir

- Hraðvirkt fyrir mjög stór fylki
- Viðheldur fyrri röð (stable)
- Getur verið samskeiða (parallel)
- Nýtir vel skyndiminni (cache)
- Nálgast bestu mögulega tímaflækju fyrir slembið inntak
- Óhugsandi að tímaflækjan sé verri en $O(n \log n)$

Gallar

- Hægvirkt fyrir lítil fylki
- Græðir ekki á fyrri röð
- Krefst hjálparfylkis
- Ekki auðveldlega „superscalar“
- Erfiðara að gera samskeiða en samplesort

Multiway mergesort

Sorting based on merging multiple sorted sequences

Basic idea in multiway mergesort

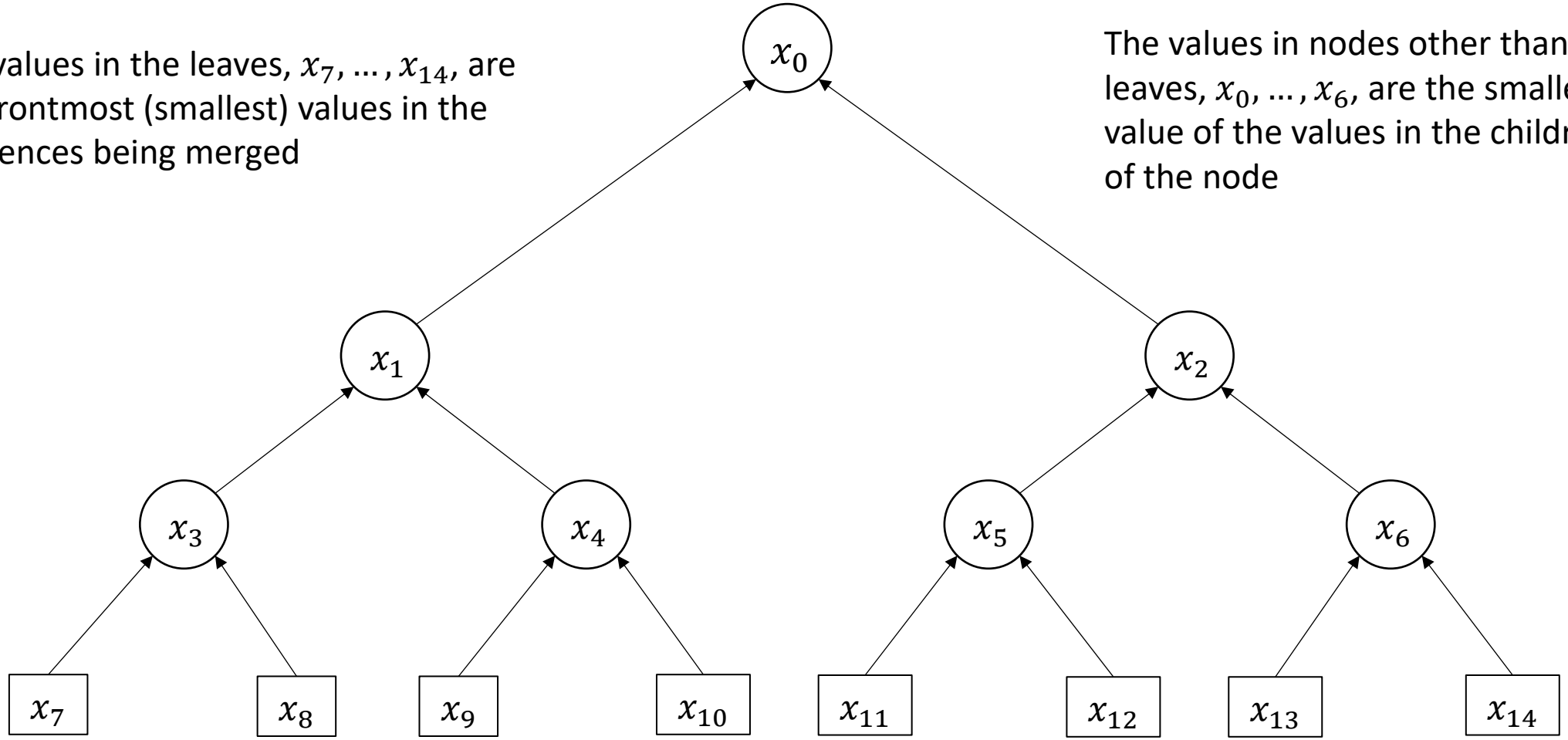
- Split the bag of values being sorted into k equally big bags $S_1, \ldots, S_k$
- Sort each bag S_i yielding sequence R_i
- Merge the sequences yielding a sorted total sequence

Merge tree (knockout competition of values)

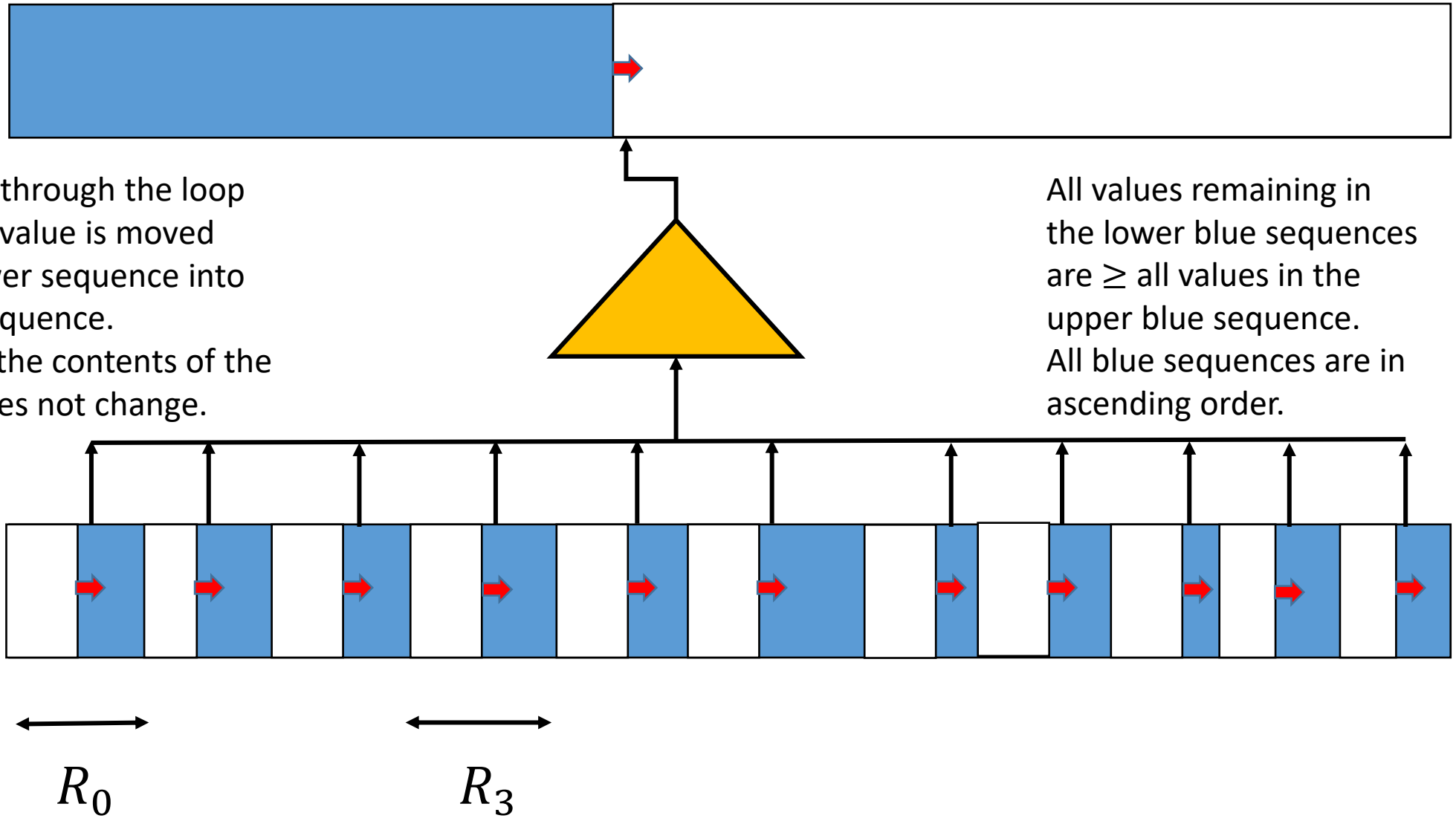
Sometimes called a tournament tree

The values in the leaves, $x_7, \ldots, x_{14}$, are the frontmost (smallest) values in the sequences being merged

The values in nodes other than leaves, $x_0, \ldots, x_6$, are the smaller value of the values in the children of the node



Multisequence merge loop



In each pass through the loop the smallest value is moved from the lower sequence into the upper sequence. Collectively, the contents of the blue area does not change.

All values remaining in the lower blue sequences are ≥ all values in the upper blue sequence. All blue sequences are in ascending order.

Advantages and disadvantages of multiway mergesort

Advantages

- Fast for very large arrays
- Stable
- Can be parallel
- Cache friendly
- Approaches best possible time complexity for random inputs
- Can not be worse than $O(n \log n)$

Disadvantages

- Slow for small arrays
- Does not profit from existing order
- Needs helper array
- Not easily superscalar
- Harder than samplesort to parallelize