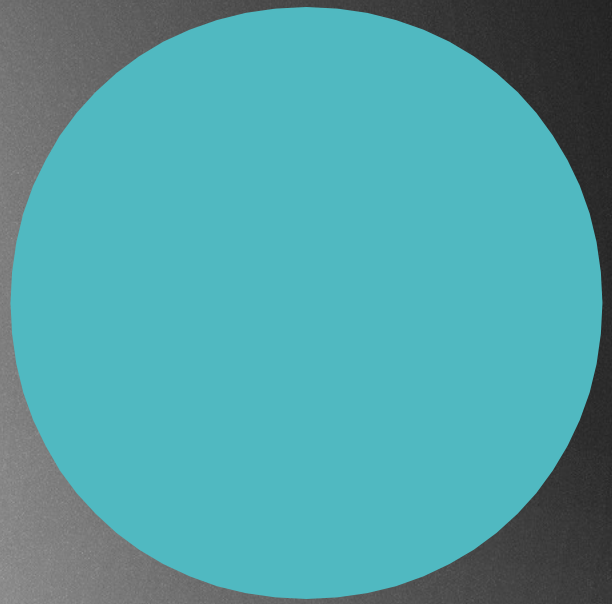
DATA STRUCTURES

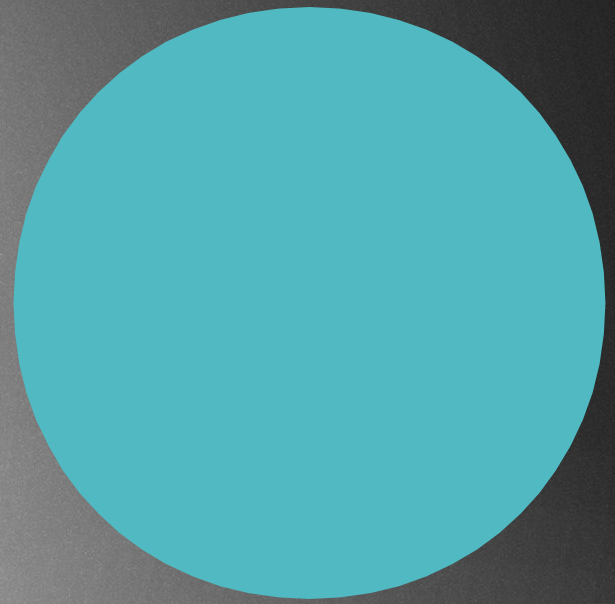


- Data structures: to store data in an efficient way
- Why to use data structures?
- We often have the intuition → if we want to make an algorithm fast, we have to optimize it
- Avoid nested for loops
- Make every calculation as fast as possible
- BUT algorithms can be boosted up by proper data structures
- Data structures make sure the running time will be better

- **Dijkstra algorithm**
- Without a proper data structure (heap / priority queue) the running time would be quadratic // **$O(N^2)$**
- Priority queue approach makes sure the running time will be far better // **O(N*logN)**

- **Spanning trees**
- We can boost up the algorithm with the help of priority queues as well
- So the running time of the algorithms are determined by the data structures we use !!!

ADT



Abstract data types

Abstract data types

- Basically this is the model (logical description) for a certain data structure
- It is like a supertype in programming (so an interface in Java)
- We just define what methods / functions the data structure will have, so we define the basic behavior
- IMPORTANT: it is just the model, ADT does not specify the concrete implementation or programming language
- „basically what the user knows”
- For example: stack → push() pop() peek()

Data structure

- The concrete implementation, the actual representation of the data
- The aim is to be able to store and retrieve data in an efficient manner
- What we want: to be able to insert / find items in **O(1)** time complexity and to be able to retreive items in **O(1)** as well
- For example: arrays, linked lists, binary trees ...

abstract data types	data structures
Stack	array, linked list
Queue	array, linked list
Priority queue	heap
Dictionary / hashmap	array