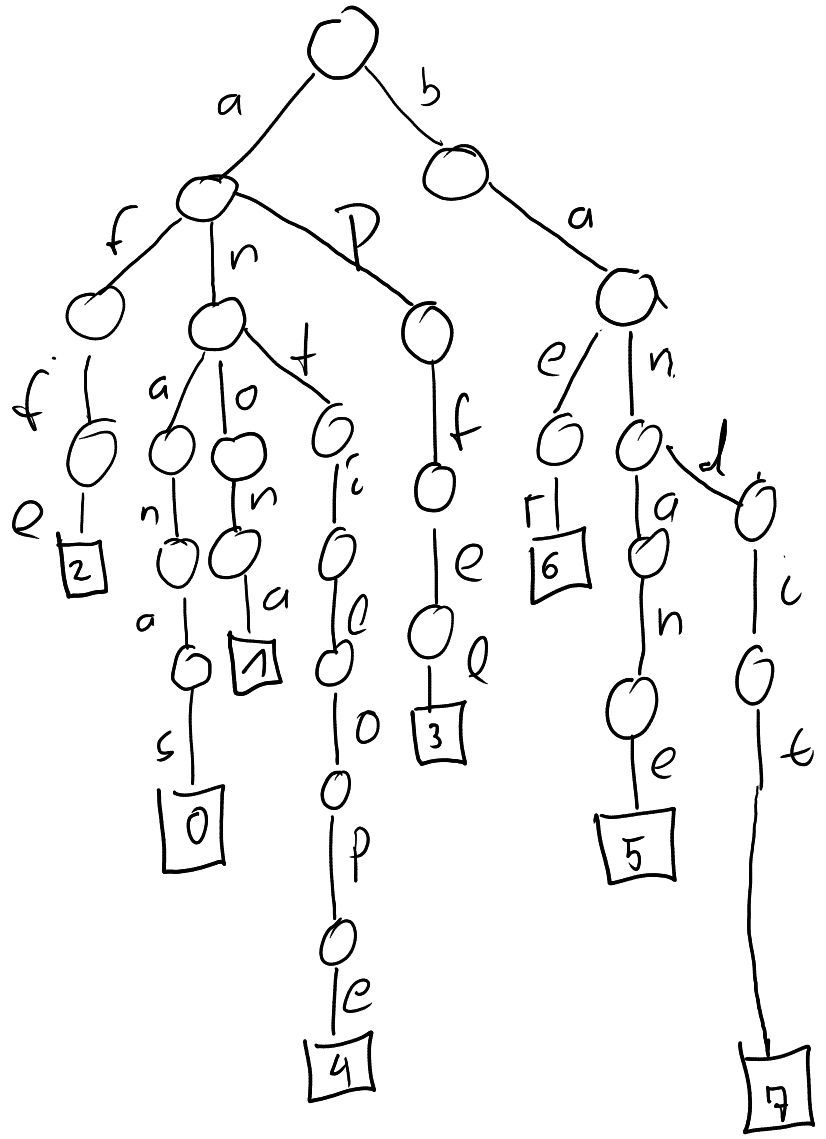


$T = T[0] \ldots T[n-2]\$$, $\$ < \alpha, \alpha \in \Sigma$ und $\$ \notin \Sigma$

$S_i' = S_i \$_i$

Habe: Menge $S = \{s_1, \dots, s_k\}$ von präfixfreien Strings mit $N := \sum |s_i|$.

Sortiere Strings mithilfe eines Tries und Arrays der Größe σ in jedem Knoten

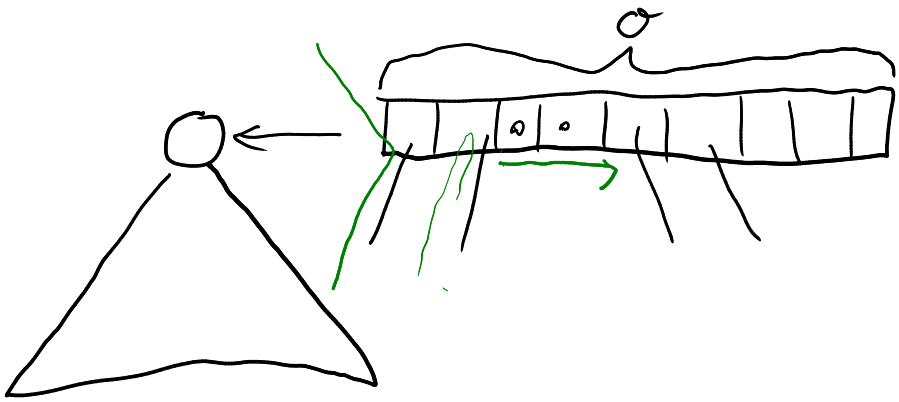
↳ Zugriff auf Kinder in $O(1)$ Zeit

↳ $O(N)$ Zeit um alle Strings einzufügen

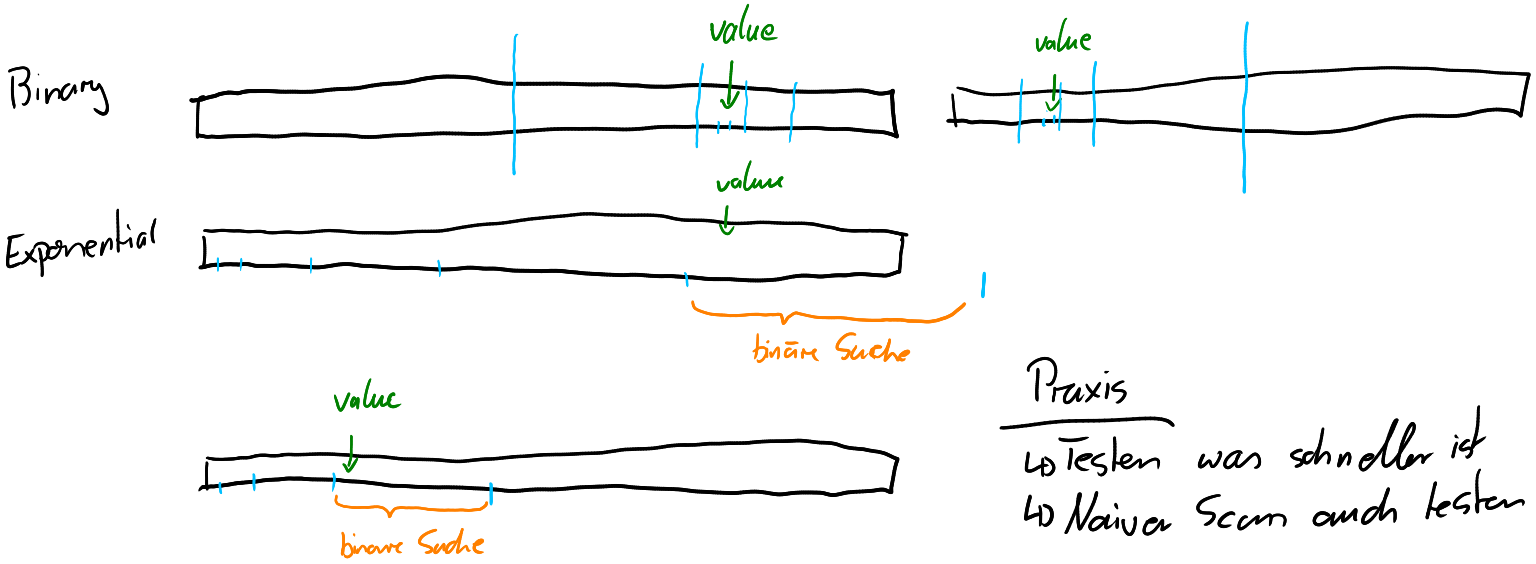
↳ In jedem Knoten müssen wir bis zu σ Kinder betrachten

↳ Es gibt $O(N)$ Knoten

↳ Um alle Strings in sortierter Reihenfolge auszugeben brauchen wir $O(N\sigma)$ Zeit.



↳ Strenggenommen braucht die Initialisierung schon $O(N\sigma)$ Zeit, aber hierfür gibt es einen Trick. → Array müssen nicht genullt werden



Praxis

- Testen was schneller ist
- Naiven Scan auch testen

Suche in sort. Arrays

Aber: std::vector

- std::array
  - Feste Größe
  - zur Kompilierzeit
- std::vector
  - Dynamische Größe
  - Kann zur Laufzeit angepasst werden

} Konsekutiver Speicher

- std::unordered_map
  - Hashtabelle
  - Key-Paare
- std::map
  - Rot-Schwarz Baum
  - Sortierte Suchbaum
  - Langsam!

} kein konsekutiver Speicher

Kopieren vermeiden

```
void foo(std::vector<> data)...
```

↑ Achtung Kopie

- Dafür haben wir Pointer (*) und Referenzen (&)

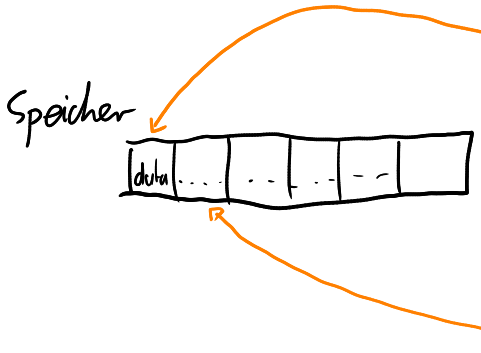
```
void foo(std::vector<>& data)...
```

Mgs... Referenz ist ein konstanter Pointer auf ein Objekt"

- Referenz ist das Objekt selber

- Pointer zeigen auf Speicherbereich

```
void foo(std::vector<>* data)...
```

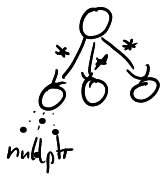


- Pointer können manipuliert werden

z.B.: `tmp = data + 1;`

- Pointer können null sein
  - z.B.: `tmp = nullptr;`

Wofür nullptr?



Was bedeutet const?

Nicht änderbarer Speich

- `size_t const count = ... // const size_t count`
- `void foo(std::vector<> const& data)...` → (innerhalb von foo kann data nicht geändert werden.
- `void foo(std::vector<> ...) const {`

→ Methode innerhalb einer Klasse kann die Klasse / Member nicht verändern

```
}
```

Lesbarkeit

- `const size_t *`
- `size_t const *`

was bedeutet das?

→ `size_t * const` → `size_t const * const`