

Gr. 1, Winkler, BSc MSc	Name _____	Aufwand in h _____
Gr. 2, Dr.Pitzer	Punkte _____	Kurzzeichen Tutor / Übungsleiter _____ / _____

Beispiel	L Lösungsidee	I Implementierung	T Tests	S = L+I+T	M Multiplikator	S*M
a	0..3	0..4	0..3		6	
b		0..5	0..5		4	
Summe (Erfüllungsgrad) Bitte auch online ausfüllen!						

Bitte erstellen Sie für diese Übung ein ZIP-Archiv in dem im Unterverzeichnis „doc“, die gesamte Dokumentation (inkl. Projektbeschreibung, Lösungsidee und formatiertem Quelltext) als PDF vorliegt, sowie der Quelltexte in den Angegebenen Unterverzeichnissen des Verzeichnis „src“.

Denken Sie bitte auch an die Angabe des Aufwandes auf dem Deckblatt und online!

Skip Lists

Zur Realisierung der abstrakten Datenstrukturen sortierte Menge (*set*) und sortiertes assoziatives Feld (*dictionary*, *map*) werden meistens binäre Suchbäume verwendet. Damit die Operationen auf diese Datenstrukturen (Finden, Einfügen und Löschen) in allen Fällen eine logarithmische Laufzeitkomplexität aufweisen, muss der Suchbaum bei jeder schreibenden Operation ausbalanciert werden. Die Implementierung von balancierten Bäumen (Rot-Schwarz-Baum) ist allerdings sehr aufwändig.

In einer schon relativ alten Arbeit aus dem Jahr 1990 stellt William Pugh sogenannte *Skip Lists* vor (das vollständige Paper finden Sie unter <https://homepage.divms.uiowa.edu/~ghosh/skip.pdf>), welche sich ebenfalls zur Implementierung der oben angesprochenen abstrakten Datenstrukturen eignen. Skip-Lists haben zwar im Durchschnitt dasselbe asymptotische Laufzeitverhalten wie balancierte Suchbäume, sind aber bei Weitem einfacher zu implementieren.

- a) Studieren Sie das Paper von Pugh und erstellen Sie auf Basis der daraus gewonnenen Erkenntnisse eine C++-Schablone *skip_set*, welche die abstrakte Datenstruktur Menge implementiert, allerdings intern Skip Lists zur Repräsentation der Knoten verwendet. Die Schnittstelle von *skip_set* soll zunächst folgendermaßen aussehen:

```
template<typename T, const int MAXLEVEL=32>
class skip_set {
public:
    skip_set();
    ~skip_set();
    int size() const;
    bool find(T value);
    void insert(T value);
    bool erase(T value);
    friend std::ostream& operator << (std::ostream &os, const skip_set &s);
};
```

- b) Führen Sie für die Mengen-Implementierung in der STL und Ihre Implementierung ausführliche Laufzeitanalysen durch. Übersetzen Sie dazu Ihre Quelltexte mit aktivierter Optimierungsoption. Überprüfen Sie zunächst, ob das erwartete asymptotische Laufzeitverhalten tatsächlich eintritt und belegen Sie Ihre Beobachtungen mit einer Grafik. Präzisieren Sie die Unterschiede im Laufzeitverhalten, indem Sie aus Ihrem Datenmaterial eine Regressionsfunktion für beide Implementierungen ermitteln.