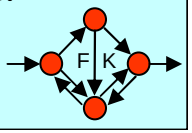
7.4.4: Prozedur für das Absinken

```
procedure sink (links, rechts: 1..n) is          -- A und n sind global
i, j: Natural;  weiter: Boolean:=true;  v: <Elementtyp>;
begin  v := A(links);  i := links;  j := i+i;
    while  (j <= rechts) and weiter  loop
        if  j = rechts  then
            if A(j) > v then A(i):=A(j); i:=j; end if;
            weiter:=false;
        elsif A(j) < A(j+1) then
                if v < A(j+1) then  A(i) := A(j+1); i := j+1;
                else weiter:=false; end if;
        else    if v < A(j) then A(i) := A(j); i := j;
                else weiter:=false; end if;
        end if;
        j := i+i;                    -- v wird erst am Ende explizit eingetragen.
    end loop;                        -- i gibt am Ende die aktuelle Position an, an
    A(i):=v;                         -- der v einzufügen ist.
end sink;                            -- Im Inneren der Schleife werden bis zu zwei
                                     --Vergleiche zwischen Elementen durchgeführt.
```



Definition 6.3.11: Kollisionsstrategien (Fortsetzung)

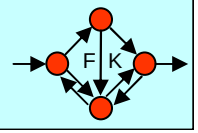
Es seien f und g zwei unterschiedliche Hashfunktionen. Sei i die Zahl der Zugriffe (beginnend mit i = 0). Die Kollisionsstrategie

$D(s,i) = (f(s) + i \cdot g(s)) \bmod p$ heißt "Doppel-Hash-Verfahren".

Es seien f_1, f_2, f_3, f_4, ... eine Folge von möglichst unterschied-lichen Hashfunktionen. Die Kollisionsstrategie

$M(s,i) = f_i(s)$ heißt "Multi-Hash-Verfahren".

Hinweis: In der Praxis hat man mit Doppel-Hash-Strategien gute Erfahrungen gemacht.

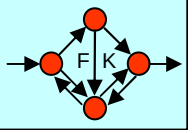


6.3.12 Löschen in Hashtabellen (DELETE)

Dieses bildet das Hauptproblem beim offenen Hashing.

Der einfachste Weg ist es, das Löschen durch Setzen eines Booleschen Wertes zu realisieren: Wenn der Eintrag mit dem Schlüssel s gelöscht werden soll, so suche man seine Posi-tion j auf und setze A(j).geloescht := true.
Beim Einfügen behandelt man dieses Feld A(j) dann wie einen freien Platz (nicht aber beim Suchen).

Nachteil: Wenn oft gelöscht wird, dann ist die Tabelle schnell voll und muss mit gewissem Aufwand reorganisiert werden, vgl. Abschnitt 6.5. Dennoch ist dieses Vorgehen in der Praxis gut einsetzbar.



Hat man sich jedoch für das lineare Sondieren entschieden, dann kann man das Löschen korrekt durchführen: Man sucht den Eintrag A(j) mit dem zu löschenden Element auf und geht dann die Einträge A(j+c), A(j+2c), A(j+3c) solange durch, bis man auf eine freie Komponente stößt. In dieser Kette kopiert man alle Einträge um c, 2c, 3c usw. Plätze zurück, aber niemals über die Komponente k hinaus mit f(s)=k.

Details: selbst überlegen! Siehe auch Übungen.