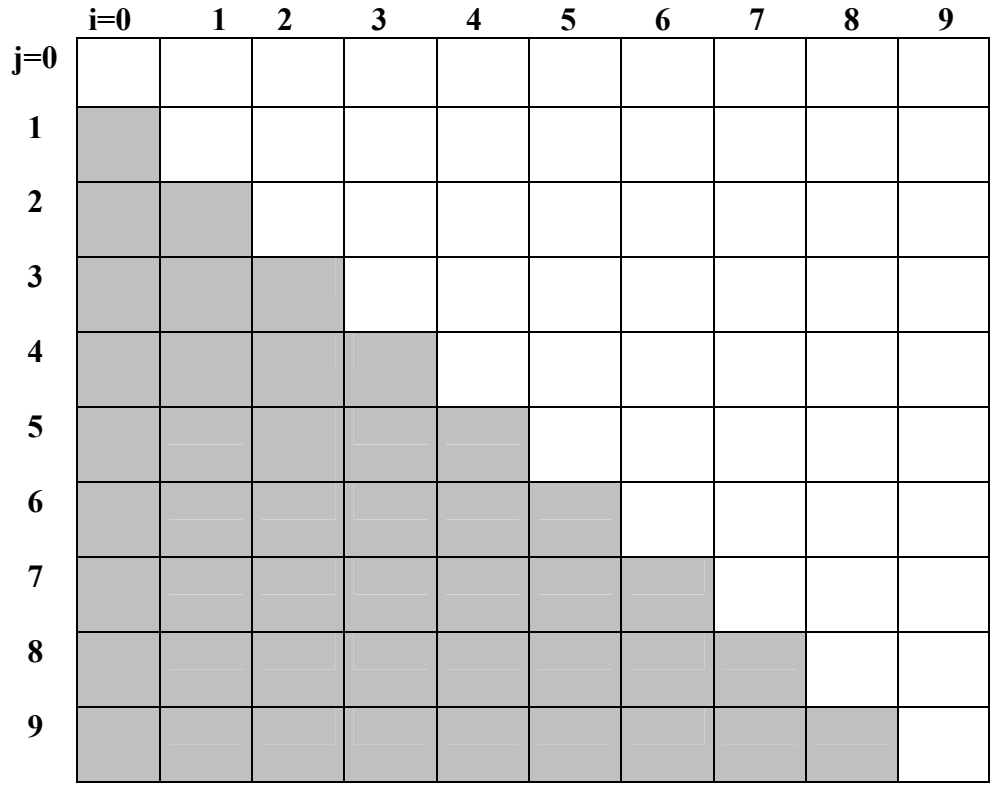
ALPIII Übung 4 24.Nov.2003

Name: Mie Rie Lim-Becker
Yao Chen
Tutor: Till Zoppke

Aufgabe 24 b

24. (3 Punkte) Speichern ungeordneter Paare in einem Array.
(a) (0 Punkte) Wie kann man die Lösung der vorigen Aufgabe dazu verwenden, für eine feste Liste von n Objekten die Paare, die man aus je zwei dieser Objekte bilden kann, möglichst kompakt zu verwalten und zu speichern (wie in einer Hash -Tabelle), sodass man in konstanter Zeit auf jedes Paar {u, v} zugreifen kann, wenn man u und v kennt?
(b) (3 Punkte) Wie kann man dieses Problem auf andere Art mit einem Array der (optimalen) Länge n (n + 1)/2 lösen?



u und v sind die Objekte. Einem Objekt-Paar {u,v} kann man also ein Zahlen-Paar(i,j) mit l<=i<j<=n zuordnen. Diese Paare kann man in einem Diagramm darstellen, wie oben gezeigt. Der graue Bereich ist der Bereich, der alle Paare (i,j) mit i<j enthält.

Alle diese Paare sollte in einem Array (Länge (n (n-1))/2) gespeichert werden(Wie in einer Hash Tabelle), Über diese Paare nehmen wir nun eine zeilenweise Abzählung vor:

(0,1)
(0,2), (1,2)
(0,3), (1,3), (2,3)
...

(0,n-1), (1,n-1), ... , (n-2,n-1)

(0,1)	(0,2)	(1,2)	(0,3)	(1,3)	(2,3)	…	(0,n-1)	(1,n-1)	(1,n-2)	…	(n-2,n-1)

Daraus ergibt sich folgende Summenformel, die in einem weiteren Schritt in eine geschlossene Formel umgewandelt wird.

$$H(i,j) = \left(\sum_{P=1}^{j-1} p\right) + i$$

aufgelöst wie folgt: = ((j-1)*j)/2 + i

Da wir diese Formel durch eine Abzählung konstruiert haben, brauchen wir die Injektivität nicht nachzuweisen.

Aufgabe 28 (5 Punkte)

Entfernen Sie die Endrekursion aus der Methode zugroß im Programm zur Verwaltung einer Halde (Siehe Aufgabe 21 und das Heapsort -Programm aus der Vorlesung: http://www.inf.fu-berlin.de/~rote/Lere/2003-04-WS/Algorithmen+Programmierung3/heapsort.java)

```
void zugroß (int i) // a[i] ist möglicherweise größer als seine
Nachfolger.
{  /*  int kleinsterNachfolger;
    if (2*i+1 <= n) {
        if (a[2*i]<a[2*i+1]) kleinsterNachfolger = 2*i;
        else                          kleinsterNachfolger = 2*i+1;
    }
    else if (2*i <= n) kleinsterNachfolger = 2*i;
    else return;
    if (a[i] > a[kleinsterNachfolger]) {
        vertausche (i, kleinsterNachfolger);
        zugroß(kleinsterNachfolger);
    }

//Rekursion auslösen

void zugroß (int i) // a[i] ist möglicherweise größer als seine
Nachfolger.
{     int kleinsterNachfolger = 0;

      while (2*i<=n) {
       if ( (2*i+1)<= n){
         if ( (a[2*i]<a[2*i+1])) {
           kleinsterNachfolger = 2*i;
         }else{
           kleinsterNachfolger = 2*i+1;
```

```
          }
        }else{
           kleinsterNachfolger = 2*i;
        }
        if (a[i] > a[kn]){
         vertausche (i, kn);
         i=kleinsterNachfolger;
        }else{
          break;
        }
    }
}
```