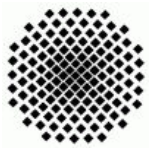


Universität Stuttgart
Fakultät 5, Institut FMI
Evolutionären Algorhitmen

Sommersemester 2006
Claus, Riexinger
03. Mai 2006

Aufgabenblatt 1 **Abgabe: 14. Mai 2006, 24.00 Uhr**

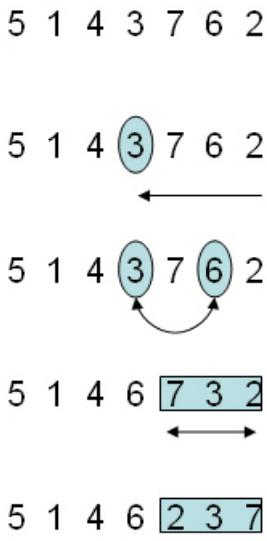
1. Die C-Funktion (leicht) (4)

Schreiben Sie eine Funktion, die für eine gegebene Permutation $P \in \Sigma^*$ der Länge n den Wert der C-Funktion $C : \Sigma^* \rightarrow \mathbb{Q}$ ermittelt .

2. Nächste Permutation (mittel) (5)

Implementieren Sie eine Funktion **nextPerm** die bei einer gegebenen Permutation P_i die nachfolgende Permutation P_{i+1} berechnet, wobei gilt $P_{i+1} > P_i$ (wenn man die Permutation als Zahl mit den Ziffern $1, 2, .., n$ auffasst. Gehen Sie dabei wie folgt vor. Gehen Sie die Elemente der Permutation von hinten nach vorne durch, bis Sie zu einem Element kommen, das kleiner als das zuvor besuchte Element ist. Vertauschen Sie nun dieses Element mit dem nächst größeren Element im "rechten" Teil der Permutation. Anschliessend spiegeln Sie die "rechts" stehenden Elemente.

Beispiel:



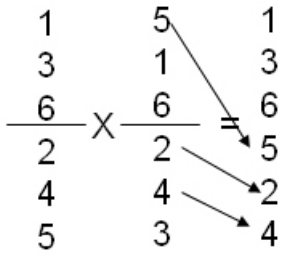
3. Suche des maximalen Wertes der C-Funktion für n Elemente (mittel) (4)

Es sei $P_{max}^{(n)}$ eine Permuation von n Elementen, deren C-Wert maximal bezüglich aller anderen solchen Permutationen ist, d.h. $C(P_{max}^{(n)}) \geq C(P), \forall P \in S_n$. Schreiben Sie nun ein Programm, das mittels der in den beiden vorherigen anderen Aufgaben entwickelten Funktionen für eine vom Benutzer zu erfragende Länge n eine Permutation $P_{max}^{(n)}$ von n Elementen ermittelt. Geben Sie Testläufe für $n = 3$ bis 9 ab.

4. Crossover-Operator (mittel) (5)

Schreiben Sie eine Funktion **crossover**, die zwei Permutationen $P_1, P_2 \in \Sigma^*$ als Eingabe erhält und diese beiden dann mittels einer zufällig zu ermittelnden Stelle rekombiniert: $X : \Sigma^* \times \Sigma^* \to \Sigma^*$. Da es sich hierbei wiederum um Permutationen $P \in \Sigma^*$ handelt, ist darauf zu achten, dass keine Elemente a_i, a_j in P doppelt vorkommen $\forall_{i \neq j} : a_i \neq a_j$. Daher geht man in der Regel wie folgt vor. Man ermittelt eine Stelle k, an der geteilt werden soll und übernimmt den ersten Teil des ersten Individuums. Die Elemente im verbleibenden Teil des Individuums ordnet man nun gemäß der Reihenfolge ihres Vorkommens im zweiten Individuum an.

Beispiel ($k = 3$):



5. Mutations-Operator (leicht) (2)

Implementieren Sie eine Funktion **mutation** $M : \Sigma^* \to \Sigma^*$, die eine Permutation als Eingabe erhält. Diese Funktion ermittelt zwei zufällige Stellen j und k innerhalb der gegebenen Permutation, die dann vertauscht werden. Also für j und k: $M(a_1 a_2..a_n) = a_1 a_2..a_{j-1} a_k a_{j+1}..a_{k-1} a_j a_{k+1}..a_n$ (beachte $j = k$ ist erlaubt).

6. Alternativ zu Aufgabe 2 und 3: Evolutionäres Vorgehen zur Ermittlung eines maximalen Wertes der C-Funktion (mittel) (10)

Schreiben Sie ein Programm, das den in Kapitel 3 des Skriptes vorgestellten Basisalgorithmus umsetzt. Gestalten Sie den Algorithmus so, dass er für eine gegebene Länge n evolutionär die Permutation ermittelt, für die der Wert der C-Funktion möglichst gross ist. Untersuchen Sie hierbei verschiedene Einstellungen der Parameter und erstellen Sie eine kurze Statistik Ihrer Ergebnisse.

Anmerkungen:

S_n = Menge der Permutationen der ersten n Zahlen. Eine Permutation $P \in S_n$, $P = (1 \to P_1, 2 \to P_2, .., n \to P_n)$ wird in der Regel als Wort $P_1 P_2..P_n$ geschrieben, d.h. $P \in \Sigma^*$ mit $\Sigma = 1, 2, .., n$ und alle Zahlen treten in P genau einmal auf.

Pro Aufgabenblatt werden 20 Punkte angerechnet.