DGA Ü3

1.) Sei S ein (unsortiertes) Feld von ganzen Zahlen und x eine ganze Zahl. Gesucht ist ein Algorithmus, der feststellt ob für zwei Zahlen a, b aus S gilt $x = a + b$, mit worst-case Laufzeit von $\Theta(n \log n)$. $n = len(S)$

Idee: zuerst S sortieren, dann von links nach rechts die Zahlen durchgehen für den ersten Summanden a. Um b zu finden (oder eben nicht) mittels binary-search den rechts von a liegenden Teil des Feldes durchsuchen.

Algorithmus: Aufwand:

```
is_sum_in_field(S, x):
    n = length(S);                       1
    S.sort();                            n·log(n)
    for a = 0,...,n:                     n
        l = a+1; r = n-1;                n
        while l < r:                     n·log_2(n-a)
            m = ⌊(l+r)/2⌋                n·log_2(n-a)
            if (S[a] + S[m]) == x:       n·log_2(n-a)
                return true;
            else if (S[a] + S[m]) < x:   n·log_2(n-a)
                l = m+1;
            else:                        n·log_2(n-a)
                r = m-1;                 n·log_2(n-a)
    return false;                        1
```

$\Rightarrow$ insgesamter Aufwand $\Theta(n \log(n))$

DGA Ü3

3) a) $T(n) = 9\,T(\frac{n}{3}) + n^2$ $\quad a=9 \quad b=3 \quad f(n)=n^2$

$$f(n) = n^2 = \Theta(n^2) = \Theta(n^{\log_3(9)})$$

$$\Rightarrow T(n) = \Theta(n^{\log_3(9)} \log(n)) = \Theta(n^2 \log(n))$$

b) $T(n) = 8\,T(\frac{n}{2}) + n!$ $\quad a=8 \quad b=2 \quad f(n)=n!$

$$n^{\log_b a+\varepsilon} = n^{\log_2 8+\varepsilon} = O(n^{\log_2 8}) = O(n^3) = O(n!) = O(f(n))$$

$$a \cdot f(\tfrac{n}{b}) = 8 \cdot f(\tfrac{n}{2}) = 8 \cdot (\tfrac{n}{2})! \leq \tfrac{1}{2}\, n! = \tfrac{1}{2} \cdot f(n)$$

also $f(n) = \Omega(n^{\log_b a+\varepsilon}) \wedge a f(\frac{n}{b}) \leq c f(n)$ für ein $c<1$

$$\Rightarrow T(n) = \Theta(f(n)) = \Theta(n!)$$

c) $T(n) = T(\frac{n}{\frac{5}{4}}) + n \log(n)$ $\quad a=1 \quad b=\frac{5}{4} \quad f(n)=n\log(n)$

$$n^{\log_b(a+\varepsilon)} = n^{\log_{\frac{5}{4}}(1+\frac{1}{4})} = n = O(n \log(n))$$

$$a \cdot f(\tfrac{n}{b}) = 1 \cdot f(\tfrac{4}{5} n) = \tfrac{4}{5} n \log(\tfrac{4}{5} n) = \tfrac{4}{5} n (\log(n) + \log(\tfrac{4}{5}))$$

$$= \tfrac{4}{5} f(n) + \tfrac{4}{5} n \log(\tfrac{4}{5}) \approx \tfrac{4}{5} f(n) - 0{,}1785\, n \leq \tfrac{4}{5} f(n)$$

$$\Rightarrow T(n) = \Theta(n \log(n))$$

d) $T(n) = 5\,T(\frac{n}{2}) + \log(n+1)$ $\quad a=5 \quad b=2 \quad f(n)=\log(n+1)$

$$n^{\log_b a-\varepsilon} = n^{\log_2(5-1)} = n^2 \qquad f(n) = \log(n+1) = O(n^2)$$

$$\Rightarrow T(n) = \Theta(n^{\log_2 5})$$

e) $T(n) = T(\frac{8}{9} n) + n$ $\quad a=1 \quad b=\frac{9}{8} \quad f(n)=n$

$$n^{\log_b(a+\varepsilon)} = n^{\log_{\frac{9}{8}}(1+\frac{1}{8})} = n = O(n) = O(f(n))$$

$$a \cdot f(\tfrac{n}{b}) = f(\tfrac{8}{9} n) = \tfrac{8}{9} n \leq \tfrac{8}{9} n = \tfrac{8}{9} f(n)$$

$$\Rightarrow T(n) = \Theta(f(n)) = \Theta(n)$$

f) $T(n) = 11\,T(\frac{n}{3}) + n^{1,5}$ $\quad a=11 \quad b=3 \quad f(n)=n^{1,5}$

$$n^{\log_b(a-\varepsilon)} = n^{\log_3(11-2)} = n^2 \qquad f(n) = n^{1,5} = O(n^2)$$

$$\Rightarrow T(n) = \Theta(n^{\log_3 11})$$

DGA Ü3

4.) a) A... (kn×n) Matrix B... (n×kn) Matrix

ges: Aufwand von $A \cdot B$ mit Strassen Algorithmus als Unterprogramm

```
Algorithm(A,B):
    C = n×n-Matrix mit Nullern;                    // n^2
    for i=0,...,k:
        Ã = i-te (n×n) Teilmatrix von A;
        B̃ = i-te (n×n) Teilmatrix von B;
        C̃ = Strassen_Algorithmus(Ã, B̃);           // k·n^(log2 7)
        for j=0,...,n:
            for k=0,...,n:
                C[j,k] = C[j,k] + C̃[j,k];          // n^2·k
    return C;
```

$\Rightarrow$ Aufwand von $n^2 + k \cdot n^{\log_2 7} + n^2 \cdot k$

b) ges: Aufwand von $B \cdot A$ mit Strassen Algorithmus als Unterprogramm

```
Algorithm(B,A):
    C = kn×kn-Matrix mit Nullern;                  // k^2 n^2
    for i=0,...,k:
        for j=0,...,k:
            Ã = i-te (n×n) Teilmatrix von A;
            B̃ = j-te (n×n) Teilmatrix von B;
            C̃ = Strassen_Algorithmus(B̃, Ã);       // k^2·n^(log2 7)
            for l=0,...,n:
                for m=0,...,n:
                    C[i·k+l, j·k+m] = C̃[l,m];      // k^2·n^2
    return C;
```

$\Rightarrow$ Aufwand von $k^2 n^2 + k^2 \cdot n^{\log_2 7}$

DGA Ü3

5.) $M=(ab)_{10}=10a+b \qquad N=(cd)_{10}=10c+d$

$A=ac \qquad B=bd \qquad C=(a-b)(d-c)$

$\Rightarrow MN=(ab)_{10}\cdot(cd)_{10}=100A+10A+10B+B+10C$

a) zz: $(ab)_{10}\cdot(cd)_{10}=100A+10A+10B+B+10C$

$100A+10A+10B+B+10C=100ac+10ac+10bd+bd+10(a-b)(d-c)$

$=100ac+\cancel{10ac}+\cancel{10bd}+bd+10ad-\cancel{10ac}-\cancel{10bd}+10bc$

$=10a(10c+d)+b(10c+d)=(10c+d)(10a+b)=(ab)_{10}\cdot(cd)_{10}$

b) ges: ähnlicher Algorithmus für n-stellige Zahlen

$M=10^{\lfloor\frac{n}{2}\rfloor}a+b \qquad N=10^{\lfloor\frac{n}{2}\rfloor}c+d$

$A=ac \quad B=bd \quad C=(a-b)(d-c)$

$\Rightarrow MN=10^{2\lfloor\frac{n}{2}\rfloor}A+10^{\lfloor\frac{n}{2}\rfloor}A+10^{\lfloor\frac{n}{2}\rfloor}B+B+10^{\lfloor\frac{n}{2}\rfloor}C$

$=10^{2\lfloor\frac{n}{2}\rfloor}ac+\cancel{10^{\lfloor\frac{n}{2}\rfloor}ac}+\cancel{10^{\lfloor\frac{n}{2}\rfloor}bd}+bd+10^{\lfloor\frac{n}{2}\rfloor}(ad-\cancel{ac}-\cancel{bd}+bc)$

$=10^{2\lfloor\frac{n}{2}\rfloor}ac+bd+10^{\lfloor\frac{n}{2}\rfloor}ad+10^{\lfloor\frac{n}{2}\rfloor}bc$

$=10^{\lfloor\frac{n}{2}\rfloor}a(10^{\lfloor\frac{n}{2}\rfloor}c+d)+b(10^{\lfloor\frac{n}{2}\rfloor}c+d)$

$=(10^{\lfloor\frac{n}{2}\rfloor}a+b)(10^{\lfloor\frac{n}{2}\rfloor}c+d)=MN$

```
Algorithm(M, N, n):
    if n == 1: return M * N;
    a = M // 10**(n/2);   b = M % 10**(n/2);
    c = N // 10**(n/2);   d = N % 10**(n/2);
    A = Algorithm(a, c);  B = Algorithm(b, d);
    C = Algorithm(a-b, d-c);
    return 10**n * A + 10**(n/2) * A + 10**(n/2) * B + B + 10**(n/2) * C;
```

c) Angenommen $n=2^k, k\in\mathbb{N}$ $\quad T(n)$... Anzahl einstelliger Multiplikationen von zwei n-stelligen Zahlen. $T(1)=1 \qquad T(n)=3T(\frac{n}{2})+4 \qquad a=3 \quad b=2 \quad f(n)=4$

$n^{\log_b(a-\varepsilon)}=n^{\log_2(3-1)}=n \qquad f(n)=4=O(n) \Rightarrow T(n)=\Theta(n^{\log_2 3})$

DGA Ü3

6.) zz: Algorithmus gibt alle m-elementigen Teilmengen mit gleicher Wahrscheinlichkeit zurück $\Leftrightarrow$ jedes Element $x \in \{1,\dots,n\}$ hat gleiche Wahrscheinlichkeit in S zu liegen.

Bei fairer Verteilung gilt $\forall x \in \{1,\dots,n\}: P(x \in S) = \frac{\binom{n-1}{m-1}}{\binom{n}{m}}$

$$= \frac{\frac{(n-1)!}{(m-1)!(n-1-(m-1))!}}{\frac{n!}{m!(n-m)!}} = \frac{m!(n-1)!\cancel{(n-m)!}}{n!(m-1)!\cancel{(n-m)!}} = \frac{m(m-1)!(n-1)!}{n(n-1)!(m-1)!} = \frac{m}{n}$$

Zeigen wir nun, dass bei jedem rekursiven Funktionsaufruf gilt $\forall x \in \{1,\dots,n\}: P(x \in S)$ (wobei S eine m-elementige Teilmenge ist) mit vollständiger Induktion nach m:

m=0 : $\emptyset$ ist die einzige 0-elementige Teilmeng $\Rightarrow P(x \in S) = 0 = \frac{0}{n}$

m+1: Nach Induktionsannahme gilt nach dem rekursiven Funktionsaufruf $\forall x \in \{1,\dots,n\}: P(x \in S) = \frac{m}{n}$. Nach der Zeile $i := \text{Random}(1,\dots,n+1)$; und dem if-statement gilt nun

$$\forall x \in \{1,\dots,n\}: P(x \in S_{neu}) = P(x \in S_{alt}) + P(i \notin S_{neu}) \cdot \frac{1}{n}$$

$$= \frac{m}{n} + \left(1 - \frac{m+1}{n+1}\right) \cdot \frac{1}{n} = \frac{m}{n} + \frac{\cancel{n+1}-m\cancel{-1}}{n(n+1)} = \frac{mn\cancel{+m}+n\cancel{-m}}{n(n+1)}$$

$$= \frac{n(m+1)}{n(n+1)} = \frac{m+1}{n+1}$$

Für n+1 gilt $P(n+1 \in S_{neu}) = \frac{1}{n+1} \cdot P(n+1 \notin S_{neu}) + P(i \in S_{alt})$

$$= \frac{1}{n+1} \cdot 1 + \frac{m}{n+1} = \frac{m+1}{n+1}$$

$$\Rightarrow \forall x \in \{1,\dots,n+1\}: P(x \in S_{neu}) = \frac{m+1}{n+1}$$