DGA Ü8

1) $a = [0, 1, 2, 3]$

(1) RECURSIVE-FFT(a) {

(2) $n = |a|$;

(3) if $n=1$ { return a; }

(4) $\omega = e^{2\pi i/n}$; $\omega' = 1$;

(5) y_0 = RECURSIVE-FFT($[a_0, a_2, \dots, a_{n-2}]$);

(6) y_1 = RECURSIVE-FFT($[a_1, a_3, \dots, a_{n-1}]$);

(7) for $k = 0, \dots, \frac{n}{2} - 1$ {

(8) $y[k] = y_0[k] + \omega' y_1[k]$;

(9) $y[k+(n/2)] = y_0[k] - \omega' y_1[k]$;

(10) $\omega' = \omega' \omega$;

(11) }

(12) return y;

(13) }

---

(1) $a = [0,1,2,3]$

(2) $n=4$

(4) $\omega = e^{\frac{\pi i}{2}}$; $\omega' = 1$

(5) → (1) $a = [0,2]$

(2) $n=2$

(4) $\omega = e^{\pi i}$; $\omega' = 1$

(5) → (1) $a = [0]$

(2) $n=1$

(3) → $[0]$ ... y_0

(6) → (1) $a = [2]$

(3) → $[2]$ ... y_1

(7) $k=0$

(8) $y[0] = 0 + 1 \cdot 2 = 2$

(9) $y[1] = -2$

(10) $\omega' = e^{\frac{\pi i}{2}}$

(11)

(12) → $[2, -2]$ ... y_0

(6) → (1) $a = [1,3]$

(2) $n=2$

(4) $\omega = e^{\pi i}$; $\omega' = 1$

(5) → (1) $a = [1]$

(3) → $[1]$ ... y_0

(6) → (1) $a = [3]$

(3) → $[3]$ ... y_1

(7) $k=0$

(8) $y[0] = 1 + 1 \cdot 3 = 4$

(9) $y[1] = -2$

(12) → $[4, -2]$ ... y_1

(7) $k=0$

(8) $y[0] = 2 + 1 \cdot 4 = 6$

(9) $y[2] = 2 - 1 \cdot 4 = -2$

(10) $\omega' = e^{\frac{\pi i}{2}}$

(7) $k=1$

(8) $y[1] = -2 + (-2)e^{\frac{\pi i}{2}} = -2-2i$

(9) $y[3] = -2 - (-2)e^{\frac{\pi i}{2}} = -2+2i$

(11)

(12) → $[6, -2-2i, -2, -2+2i]$

(13)

DGA Ü8

2) $A(x) = 4-4x$ $\quad B(x) = 6+2x$ $\quad$ ges: $C(x) = A(x) \cdot B(x)$ mittels FFT

1.) A und B in Koeffizientendarstellung abbilden mit 2n Koeffizienten

$a = [4, -4, 0, 0]$ $\quad b = [6, 2, 0, 0]$

2.) A und B in Punkt-Wert-Darstellung umrechnen (mittels FFT)

$\hat{a} = [0, 4-4i, 8, 4+4i]$ $\quad \hat{b} = [8, 6+2i, 4, 6-2i]$

3.) C in Punkt-Wert-Darstellung berechnen

$\hat{c} = [0, 32-16i, 32, 32+16i]$

4.) C in Koeffizientendarstellung zurückrechnen

$c = [24, -16, -8, 0]$

Probe: $(4-4x)(6+2x) = 24+8x-24x-8x^2 = 24-16x-8x^2$

zu 4.) INVERSE_FFT erhält man durch folgende Änderungen am FFT Algorithmus:

- ω und ω' mit ω^{-1} und $(\omega')^{-1}$ ersetzen
- Endergebnis durch n teilen

DGA Ü8

5) $n \in \mathbb{N}$ $\quad 1 = d_1 < d_2 < \dots < d_k$ gesucht ist Summe aus $d_1, \dots, d_k$, die n ergibt mit möglich wenigen Summanden

a) $b \leq n$ bel., sodass b als Teilsumme in der optimalen Summendarstellung von n vorkommt.

$$n = \sum_{j=1}^{m} d_{f(j)} = \sum_{j=1}^{l} d_{f(j)} + \sum_{j=l+1}^{m} d_{f(j)} \quad \text{mit } m \text{ minimal}$$

$f: \{1, \dots, m\} \to \{1, \dots, k\}$, sodass $n = \sum_{j=1}^{m} d_{f(j)}$ und $b = \sum_{j=1}^{l} d_{f(j)}$

Angenommen die Darstellung von b ist nicht optimal, d.h.

$$\exists \tilde{l} < l \quad \exists g: \{1, \dots, \tilde{l}\} \to \{1, \dots, k\} : \sum_{j=1}^{\tilde{l}} d_{g(j)} = b$$

$\Rightarrow \sum_{j=1}^{\tilde{l}} d_{g(j)} + \sum_{j=l+1}^{m} d_{f(j)} = n$ und die Summe ist um $l - \tilde{l} > 0$ kürzer $\lightning$

zu $n = \sum_{j=1}^{m} d_{f(j)}$ ist optimal

b) $m(n)$ ... Anzahl der Münzen im optimalen Fall

ges: Rekursion für $m(n)$

$\forall x < 0: m(x) = +\infty \quad ; \quad m(0) = 0 \quad ; \quad \forall x > 0: m(x) = \min\{m(x - d_l) : l \in \{1, \dots, k\}\} + 1$

c) ges: Algorithmus, der optimale Summe sucht

```
Algorithm(n) {
    if (n<0){return +∞;}
    if (n==0){return 0;}
    min = -∞;  min_l = -∞;
    for l = 1,...,k {
        tmp = Algorithm(n-d_l);
        if (tmp<min) {min=tmp; min_l = l;}
    }
    print(min_l+1); return min;
}
```

Aufwand ist n^k, da es in jedem der maximal n Schritte k Münzen "zur Auswahl" gibt. Maximal n Schritte, da wenn man die Münze $d_1 = 1$ wählt n Rekursionsaufrufe benötigt.

DGA Ü8

6) $m \times n$ Gitter Ziel Weg von $(1,1)$ nach (m,n) mit geringsten Kosten

a) zz: optimale Teilstrukturеigenschaft wird erfüllt

Sei (j,k) ein Punkt auf dem optimalen Weg von $(1,1)$ nach (m,n)

Angenommen der Weg von (j,k) nach (m,n) wäre nicht optimal. Zusammengefasst gilt also

$w((1,1),(m,n)) = k$ sind die minimalen Kosten von $(1,1)$ nach (m,n)

Sei y die Kosten ab dem Punkt (j,k), da der Weg von (j,k) nach (m,n) nicht optimal ist $\exists$ Weg von (j,k) nach (m,n) mit Kosten $x < y$

$\Rightarrow$ neuer Weg von $(1,1)$ bis (j,k) wie in originalen Weg mit Kosten $k - y$ und von (j,k) nach (m,n) mit neuem Weg und Kosten x

$\Rightarrow w$ war nicht optimal $\lightning$ $\Rightarrow$ Weg (j,k) nach (m,n) ist optimal

b)
$$k(i,j) = \begin{cases} A(m,n) & \text{falls } i=m \wedge j=n \\ A(m,j+1)+k(m,j+1) & \text{falls } i=m \\ A(i+1,n)+k(i+1,n) & \text{falls } j=n \\ \min(A(i+1,j)+k(i+1,j),\ A(i,j+1)+k(i,j+1)) & \text{sonst} \end{cases}$$

c)
```
Algorithm(i, j) {
    if(i==m ∧ j==n){return A(m,n);}
    if(i==m){return A(m,j+1)+Algorithm(m,j+1);}
    if(j==n){return A(i+1,n)+Algorithm(i+1,n);}
    up:= A(i+1,j)+ Algorithm(i+1,j);
    right:= A(i,j+1)+ Algorithm(i,j+1);
    return min(up, right);
}
```

Der Algorithmus führt Berechnungen für jeden der $\frac{(m+n)!}{m!\,n!}$ möglichen Wege durch das Gitter durch $\Rightarrow$ Aufwand $\frac{(m+n)!}{m!\,n!}$ (erfüllt die Rekursion $A(i,j) = A(i-1,j) + A(i,j-1)$ sowie $A(0,0)=1$ und $A(0,j)=1$).