1 Integrali

1. $\int x^a\,dx = \begin{cases} \frac{x^{a+1}}{a+1} + C & a \neq -1 \\ \ln|x| + C & a = -1 \end{cases}$
2. $\int \ln x\,dx = x\ln x - x + C$
3. $\int \frac{1}{\sqrt{x}}\,dx = 2\sqrt{x} + C$
4. $\int e^x\,dx = e^x + C$
5. $\int a^x\,dx = \frac{a^x}{\ln a} + C$
6. $\int \cos(ax)\,dx = \frac{sin(ax)}{a} + C$
7. $\int \sin(ax)\,dx = \frac{-cos(ax)}{a} + C$
8. $\int \tan x\,dx = -\ln|\cos x| + C$
9. $\int \frac{dx}{\cos^2 x} = \int \sec^2 x\,dx = \tan x + C$
10. $\int \frac{dx}{\sin^2 x} = \int \csc^2 x\,dx = -\cot x + C$
11. $\int \frac{1}{\sqrt{1-x^2}}\,dx = \arcsin x + C$
12. $\int \frac{dx}{ax+b} = \frac{1}{a} ln|ax+b| + C$
13. $\int \frac{1}{x^2+1}\,dx = arctan x + C$
14. $\int \frac{dx}{x^2+a^2} = \frac{1}{a} arctan\frac{x}{a} + C$
15. $\int \frac{f'(x)}{f(x)}\,dx = ln|f(x)| + C$

2 Bounds

- $\Theta(g) = \{f; \exists c_1, c_2, n_0 > 0, \forall n > n_0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$
- $\mathcal{O}(g) = \{f; \exists c, n_0 > 0, \forall n > n_0 : 0 \leq f(n) \leq cg(n)\}$
- $\Omega(g) = \{f; \exists c, n_0 > 0, \forall n > n_0 : 0 \leq cg(n) \leq f(n)\}$
- $o(g) = \{f; \forall c > 0, \exists n_0 > 0, \forall n > n_0 : 0 \leq f(n) < cg(n)\}$
- $\omega(g) = \{f; \forall c > 0, \exists n_0 > 0, \forall n > n_0 : 0 \leq cg(n) < f(n)\}$

2.1 Properties

- transitivity $f \in \Theta(g) \wedge g \in \Theta(h) \Rightarrow f \in \Theta(h)$ (for all bounds)
- reflexivity $f \in \Theta(f)$ (for Θ, $\mathcal{O}$ and Ω)
- symmetry $f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$
- transpose symmetry $f \in \mathcal{O}(g) \Leftrightarrow g \in \Omega(f)$
  $f \in o(g) \Leftrightarrow g \in \omega(f)$

2.2 Simplified Masters

- $T(n) = aT(\frac{n}{b}) + \Theta(n^d)$
  $a \geq 1; b > 1; d \geq 0$
- $a > b^d \rightarrow T(n) = \Theta(n^{\log_b a})$
- $a = b^d \rightarrow T(n) = \Theta(n^d \log_b n)$
- $a < b^d \rightarrow T(n) = \Theta(n^d)$

2.3 Masters

- $T(n) = aT(\frac{n}{b}) + f(n)$
  $a \geq 1; b > 1$
- $f(n) = \mathcal{O}(n^{log_b a - \epsilon}) \rightarrow T(n) = \Theta(n^{\log_b a}), \epsilon > 0$
- $f(n) = \Theta(n^{log_b a}) \rightarrow T(n) = \Theta(n^{log_b a}\log(n))$
- $f(n) = \Omega(n^{log_b a + \epsilon}) \rightarrow T(n) = \Theta(f(n)), \epsilon > 0$ and $af(\frac{n}{b}) \leq cf(n)$ for some $c < 1$ and big enough n
- case2 ext: $f(n) = \Theta(n^{log_b a} log^k(n)) \rightarrow T(n) = \Theta(n^{log_b a} log^{k+1}(n))$

2.4 Akra-Bazzi

$$T(n) = \sum_{i=1}^{k} a_i T(b_i n) + f(n), n > n_0$$

- $n_0 \geq \frac{1}{b_i}$, $n_0 \geq \frac{1}{1-b_i}$ for each i,
- $a_i > 0$ for each i,
- $0 < b_i$ for each i,
- $k \geq 1$,
- $f(n)$ is non-negative function,
- $c_1 f(n) \leq f(u) \leq c_2 f(n)$, for each u satisfying condition: $b_i n \leq u \leq n$
- $T(n) = \Theta(n^p(1 + \int_1^n \frac{f(u)}{u^{p+1}} du))$ we get p from: $\sum_{i=1}^k a_i b_i^p = 1$

2.5 Extended Akra-Bazzi

$T(n) = \sum_{i=1}^k a_i T(b_i n + h_i(n)) + f(n), n > n_0$ all of the conditions from Akra-Bazzi still hold plus: $|h_i(n)| = \mathcal{O}(\frac{n}{\log^2 n})$

2.6 Annihilators

Steps:

- Write the recurrence in operator form.
- Extract the annihilator for the recurrence.
- Factor the annihilator (if necessary).
- Extract the generic solution form the annihilator.
- Solve for coefficients using base cases (if known).

Operator	**Definition**
addition	$(f+g)(n) := f(n) + g(n)$
subtraction	$(f-g)(n) := f(n) - g(n)$
multiplication	$(a \cdot f)(n) := a \cdot (f(n))$
shift	$Ef(n) := f(n+1)$
k-fold shift	$E^k f(n) := f(n+k)$
composition	$(X+Y)f := Xf + Yf$
	$(X-Y)f := Xf - Yf$
	$XYf := X(Yf) = Y(Xf)$
distribution	$X(f+g) = Xf + Xg$

Operator	**Functions annihilated**
$E - 1$	α
$E - a$	αa^n
$(E-a)(E-b)$	$\alpha a^n + \beta b^n$ $(a \neq b)$
$(E-a_0)(E-a_1)\cdots(E-a_k)$	$\sum_{i=0}^k \alpha_i a_i^n$ (a_i distinct)
$(E-1)^2$	$\alpha n + \beta$
$(E-a)^2$	$(\alpha n + \beta)a^n$
$(E-a)^2(E-b)$	$(\alpha n + \beta)a^n + \gamma b^n (a \neq b)$
$(E-a)^d$	$(\sum_{i=0}^{d-1} \alpha_i n^i)a^n$

If X annihilates f, then X also annihilates Ef.
If X annihilates both f and g,
then X also annihilates $f \pm g$.
If X annihilates f, then X also annihilates αf,
for any constant α.

If X annihilates f and Y annihilates g,
then XY annihilates $f \pm g$.

3 Pseudo random generator

3.1 Linear congruential generators

$$x_i = (ax_{i-1} + c) \mod m$$

- RANDU: $x_i = 65539 x_{i-1} \pmod{2^{31}}$
- MINSTD $x_i = 16807 x_{i-1} \pmod{2^{31} - 1}$

3.2 Blum-Blum-Shrub

- $p, q \in \mathbb{P}$, large (at least 40 decimal places)
- $m = pq$
- $X_i = X_{i-1}^2 \pmod m$
- $b_i = \text{parity}(X_i)$

4 Amortized analysis

- Aggregated analysis - Determine upper bound $T(n)$ for the total cost of a sequence of n operations. Amortized cost per operation is $\frac{T(n)}{n}$.
- Accounting method - Some operations are overcharged to pay for other operations.
- Potential method $c_i' = c_i + \Phi(D_i) - \Phi(D_{i-1})$; $\Phi(D_n) \geq \Phi(D_0)$

4.1 Vsote zaporedji

- $\sum_{i=1}^n i = \frac{n(n+1)}{2}$
- $\sum_{i=m}^n z^i = \frac{z^m - z^{n+1}}{1-z}; \sum_{i=0}^{\inf} ar^i = \frac{a}{1-r}$ pri $|r| < 1$
- $H_n = \sum_{i=1}^n \frac{1}{i} = \ln n + \gamma + \frac{1}{2n}$

4.2 Parallel programming

- Amdahl $S = \frac{1}{\frac{f}{P} + (1-f)}$
- Gustafson $S_P = \frac{T_s + PT_p}{T_s + T_p}$

5 Linear programming

5.1 Standard LP

- given n real numbers $c_1, c_2, \ldots, c_n$
- m real numbers $b_1, b_2, \ldots, b_m$
- $m \times n$ real numbers a_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, n$
- we wish to find n real numbers $x_1, \ldots, x_n$ that

maximize $\sum_{j=1}^n c_j x_j$ subject to $\sum_{j=1}^n a_{ij} x_j \leq b_i, \forall i = 1, \ldots, m; x_j \geq 0$

5.2 Transformations

- $\min f(x) \to \max -f(x)$
- $a \geq b \to -a \leq -b$
- $x \in \Re \to x = x' - x''$
- $a = b \to a \leq b; -a \leq -b$

5.3 Metroplis algorithm

- If better neighbour exists, move to it.
- Otherwise choose a random neighbour, but accept better neighbours with larger probability.
- Decrease the probability of acceptance.
- In time, stohastic search turns into deterministic LS.

5.4 Simulated annealing

- Start with a random state S.
- Select random neighbour $S\prime$
- If $q(S\prime) < q(S)$, move to $S\prime$.
- Otherwise, move with probability $e^{\frac{-(q(S\prime) - q(S))}{T}}$

Decrease temperature while it's not close to zero. Usually a geometrical rule is used: $T\prime = \lambda T$, $0 < \lambda < 1$ (typically $\lambda = 0.95$)

6 Metaheuristics

6.1 Tabu search

Idea: to prevent returning back to the same local extreme, supress (parts of) solutions.

6.2 Guided local search

Metaheuristics which guide local search and helps it avoid local extremes.

- define properties (attributes) of solutions
- penalize attributes, which occur too often in local extrema
- auxiliary objective function

$$h(s) = g(s) + \lambda \cdot \sum_{i \text{ is a feature}} (p_i \cdot I_i(s))$$

Utility of punishment for property i in local extreme $s*$

$$\text{util}_i(s*) = I_i(s*) \cdot \frac{c_i}{1+p}$$

c_i is cost, p_i is current punishment for property i
In local extreme we punish the property with the largest utility (we increment p_i by 1).

6.3 Variable neighbourhood search

Idea: define several neighbourhood structures and change neighbourhood when reaching local extreme in one of them. Order neighbourhoods by the efficiency of computation.

7 Swarm intelligence

- fixed population
- autonomous individual
- communication between agents
- aggregation of similar animals, generally cruising in the same direction
- simple rules for each individual
- decentralized
- emergent behaviour

7.1 Ant colony optimization

- ants find the shortest path to food source from the nest
- they deposit pheromone along traveled path, which is used by other ants to follow the trail
- this kind of indirect communication via the local environment is called stigmergy
- adaptability, robustness and redundancy

Possible daemon actions to apply centralized actions.

7.2 Particle swarm optimization

- Individuals strive to improve themselves and often achieve this by observing and imitating their neighbours.
- Each individual has the ability to remember.
- Each particle is represented with two vectors, location and velocity.

7.2.1 Information exchange in the swarm

- historically best location x^*
- best location of informants x^+
- globally best location $x^!$

7.2.2 Moving

- Compute the fitness of each particle and update x^*, x^+ and $x^!$.
- Update the representation of particle. Velocity vector takes into account updated directions x^*, x^+ and $x^!$. Each direction is updated with some random noise.
- Move the particle in the direction of the velocity vector.