|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **logo** | | **RENCANA PEMBELAJARAN SEMESTER**  **JURUSAN TEKNIK INFORMATIKA**  **FAKULTAS TEKNOLOGI INFORMASI - ITS** | | | | | | | | | | | | |
| **MATA KULIAH** | | | | **KODE** | | **Rumpun MK** | | | **BOBOT (sks)** | | **SEMESTER** | | **Direvisi** | |
| **PERANCANGAN ANALISIS DAN ALGORITMA 2** | | | | **KI141319** | | **AP** | | | **T=3** | **P=0** | **4** | | **11 Maret 2016** | |
| **OTORISASI** | | | | **Koordinator MK** | | | **Koordinator RMK** | | | | **Ka PRODI** | | | |
|  | | |  | | | |  | | | |
| **Capaian Pembelajaran MK** | | **Program Studi** | |  | | | | | | | | | | |
| * Mampu mengidentifikasi, menganalisis dan menyelesaikan masalah secara sistematis dan terorganisasi dalam bidang teknologi informasi dan komunikasi * Mampu menggunakan dan menerapkan konsep-konsep teoritis dan empiris dalam menyelesaikan masalah di bidang teknologi informasi dan komunikasi * Mampu menganalisis kebutuhan sumber daya dalam penyelesaian masalah bidang teknologi informasi dan komunikasi * Memiliki kreativitas dalam mengembangkan berbagai alternatif solusi * Mampu berkomunikasi baik lisan dan tulisan * Bertanggungjawab pada pekerjaan sendiri | | | | | | | | | | | | |
| **Mata Kuliah** | |  | | | | | | | | | | |
| * Mahasiswa mampu secara kreatif membuat abstraksi optimal dari persoalan nyata dengan tingkat kompleksitas medium dan hard * Mahasiswa mampu mewujudkan abstraksi persoalan ke dalam suatu rancangan algoritma dengan mempertimbangkan aspek kebenaran dan efisiensi * Mahasiswa mampu menyatakan analisis kebenaran dan kompleksitas secara formal terhadap suatu rancangan algoritma * Mahasiswa dapat mengimplementasikan hasil rancangan algoritma dengan menggunakan bahasa OOP dan menguji akurasinya pada suatu online judge * Mahasiswa mampu mempresentasikan keseluruhan tahapan rancangan dan analisis algoritma secara sistematik baik secara lisan maupun tulisan | | | | | | | | | | | | |
| **Diskripsi Bahan Kajian & Topik Bahasan** | | **Bahan Kajian** | |  | | | | | | | | | | |
| Dasar-dasar algoritma dan kompleksitas, paradigma algoritma divide dan conquer, paradigma dynamic programming, paradigma greedy algorithm, beberapa struktur data lanjut berhubungan dengan dynamic programming, implementasi algoritma dynamic programming dan greedy algorithm untuk kasus graph dan string matching. | | | | | | | | | | | | |
| **Topik Bahasan** | |  | | | | | | | | | | |
| * Algoritma dan kompleksitas * Rancangan dan analisis algoritma yang memiliki paradigma divide dan conquer   + Algoritma Binary Search   + Pemrograman dinamis jenis non classical   + Algoritma Dynamic Programming   + Algoritma Greedy * Representasi beberapa struktur data lanjut yang berhubungan dengan dynamic programming   + Struktur segment tree (range min/max query, range sum query) serta teknik lazy propagation   + Fenwick Tree (binary indexed tree)   + Splay tree * Rancangan dan analisis algoritma pada struktur graph   + Minimum Spanning tree   + All pair shortest path dan single source shortest path   + Strongly connected component, topological sort dan 2-SAT problem   + Maximum flow, minimum cut, dan bipartite matching * Rancangan dan analisis algoritma problem string matching   + KMP, Boyer Moore   + Struktur data trie   + Struktur data suffix array, suffix tree, dan suffix trie | | | | | | | | | | | | |
| **Pustaka** | | **Utama :** | |  | | | | | | | | | | |
| 1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction to Algorithms Third Edition", MIT Press, 2009 | | | | | | | | | | | | |
| **Pendukung :** | |  | | | | | | | | | | |
| 1. Levitin, Anany, "Introduction to The Design & Analysis Af algorithms 3rd ed", Addison-Wesley, 2012 2. Robert Sedgewick, Kevin Wayne, Algorithms, 4th Edition, Addison Wesley, 2011 3. Stephen Halim, Felix Halim, Competitive Programming, 3rd Edition, NUS School of Computing, 2013 | | | | | | | | | | | | |
| **Media Pembelajaran** | | **Software :** | | | **Hardware :** | | | | | | | | | |
| C/C++ compiler | | | PC & LCD Projector | | | | | | | | | |
| **Team Teaching** | | Abdul Munif, Diana Purwitasari, Wijayanti Nurul Khotimah | | | | | | | | | | | | |
| **Assessment** | | Proyek Kecil , *Case-Study*, Ujian Tulis | | | | | | | | | | | | |
| **Mata Kuliah Syarat** | | Perancangan dan Analisis Algoritma 1 | | | | | | | | | | | | |
| **Mg Ke-** | **Sub-Capaian Pembelajaran MK** | | **Materi Pembelajaran** | | | **Metode / Strategi Pembelajaran** | | **Assessment** | | | | | | |
| **Indikator** | | | | **Bentuk** | | **Bobot** |
| **(1)** | Mahasiswa mampu secara kreatif membuat abstraksi optimal dari persoalan nyata dengan tingkat kompleksitas medium dan hard | | Rencana Pembelajaran, Aturan Perkuliahan  Konsep penyelesaian masalah (observasi, abstraksi, simplifikasi)  Konsensus Pseudocode | | | Kuliah Pengantar & Brainstorming,  Diskusi Kelompok | | * Ketepatan dalam membuat abstraksi penyelesaian masalah pemrograman * Ketepatan dalam mengimplementasikan aturan pseudocode dalam algoritma yang dibuat | | | | Presentasi:   * Konsep Dasar * Contoh kasus observasi, abstraksi, dan simplifikasi * Aturan-aturan pseudocode * Latihan membuat pseudocode dengan benar | |  |
| **(2)** | Mahasiswa mampu mewujudkan abstraksi persoalan ke dalam suatu rancangan algoritma dengan mempertimbangkan aspek kebenaran dan efisiensi | | Review pendekatan divide dan conquer  Problem rekursi dengan kompleksitas tinggi (Fibonacci, Maximum Sub Array, Rod Cutting Problem) | | | Kuliah  Diskusi kelompok  Latihan soal | | * Ketepatan dalam membuat relasi recurrence dari permasalahan pemrograman * Ketepatan dalam menyusun algoritma berdasarkan relasi recurrence * Ketepatan dalam menghitung kompleksitas algoritma yang dibuat | | | | Presentasi:   * Contoh kasus yang memiliki overlapping subproblem * Relasi recurrence * Implementasi relasi recurrenceke dalam algoritma | |  |
| **(3)** | Mahasiswa mampu mempresentasikan keseluruhan tahapan rancangan dan analisis algoritma secara sistematik baik secara lisan maupun tulisan | | Solusi Dynamic Programming untuk problem rekursi dengan kompleksitas tinggi  Konsep Dynamic Programming Top Down, Bottom Up | | | Kuliah  Diskusi kelompok  Latihan soal | | * Pemahaman konsep dynamic programming top down, bottom up * Kemampuan dalam mengubah solusi rekursif ke dalam solusi dynamic programming * Ketepatan dalam menyusun algoritma dan menghitung kompleksitas algoritma | | | | Presentasi   * Konsep dynamic programming top down dan bottom up * Mengubah solusi rekursif menjadi dynamic programming * Menghitung kompleksitas dynamic programming   Latihan Soal   * Fibonacci, Maximum Sub Array, Rod Cutting Problem | |  |
| **(4)** | Mahasiswa dapat mengimplementasikan hasil rancangan algoritma dengan menggunakan bahasa OOP dan menguji akurasinya pada suatu online judge | | Latihan soal-soal yang dapat diselesaikan secara rekursif dan dynamic programming | | | Latihan soal  Praktikum | | * Kemampuan dalam mengimplementasikan algoritma ke dalam kode program * Kemampuan dalam menganalisis kebenaran output program yang dihasilkan | | | | Praktek   * Latihan implementasi algoritma ke dalam kode program * Latihan submit kode program ke online judge | |  |
| **(5)** | Mahasiswa mampu mewujudkan abstraksi persoalan ke dalam suatu rancangan algoritma dengan mempertimbangkan aspek kebenaran dan efisiensi | | Evaluasi 1 | | | Ujian tulis | | * Kemampuan dalam melakukan observasi, abstraksi, dan simplifikasi * Kemampuan dalam membuat algoritma dalam bentuk rekursif, dynamic programming (top down dan bottom up) | | | | Ujian tulis (kuis 1) | | 20% |
| **(6)** | Mahasiswa mampu menyatakan analisis kebenaran dan kompleksitas secara formal terhadap suatu rancangan algoritma | | Dynamic Programming: Coin Change, Knapsack  Analisis Kompleksitas | | | Kuliah/ceramah  Diskusi kelompok | | * Kemampuan dalam mengubah solusi rekursif ke dalam solusi dynamic programming * Ketepatan dalam menyusun algoritma dan menghitung kompleksitas algoritma | | | | Presentasi   * Coin change * Knapsack * Menghitung kompleksitas algoritma | |  |
| **(7)** | Mahasiswa dapat mengimplementasikan hasil rancangan algoritma dengan menggunakan bahasa OOP dan menguji akurasinya pada suatu online judge | | Implementasi penyelesaian masalah coin change dan knapsack dengan pendekatan dynamic programming | | | Praktikum | | * Kemampuan dalam mengimplementasikan algoritma ke dalam kode program * Kemampuan dalam menganalisis kebenaran output program yang dihasilkan | | | | Praktek   * Latihan implementasi algoritma ke dalam kode program | |  |
| **(8)** | Mahasiswa mampu mewujudkan abstraksi persoalan ke dalam suatu rancangan algoritma dengan mempertimbangkan aspek kebenaran dan efisiensi | | Evaluasi Tengah Semester (ETS) | | | Ujian tulis | | * Kemampuan dalam menyusun algoritma dalam bentuk dynamic programming * Kemampuan dalam mengimplementasikan algoritma ke dalam kode program | | | | Ujian tulis atau praktek | | 30% |
| **(9)** | Mahasiswa dapat mengimplementasikan hasil rancangan algoritma dengan menggunakan bahasa OOP dan menguji akurasinya pada suatu online judge | | Pembahasan ETS  Latihan-latihan soal berbasis online judge | | | Ceramah  Diskusi kelompok | | * Kemampuan dalam mengevaluasi solusi algoritma yang telah dibuat * Kemampuan dalam menganalisis kebenaran output program yang dihasilkan | | | | Presentasi  Praktek mengimplementasikan algoritma ke dalam kode program | |  |
| **(10)** | Mahasiswa mampu secara kreatif membuat abstraksi optimal dari persoalan nyata dengan tingkat kompleksitas medium dan hard | | Teori Greedy Algorithm  Contoh kasus greedy algorithm: activity selection | | | Ceramah  Diskusi kelompok | | * Kemampuan dalam memahami konsep greedy algorithm * Kemampuan dalam menentukan greedy choice dalam kasus pemrograman | | | | Presentasi  Latihan soal | |  |
| **(11)** | Mahasiswa mampu menyatakan analisis kebenaran dan kompleksitas secara formal terhadap suatu rancangan algoritma | | Contoh kasus greedy algorithm: fractional knapsack | | | Ceramah  Diskusi kelompok | | * Kemampuan dalam menganalisis kompleksitas algoritma greedy * Kemampuan dalam menuliskan definisi formal algoritma greedy | | | | Presentasi  Latihan soal dan implementasi algoritma ke dalam kode program | |  |
| **(12)** | Mahasiswa mampu menyatakan analisis kebenaran dan kompleksitas secara formal terhadap suatu rancangan algoritma | | Huffman Codes dan implementasi | | | Ceramah  Diskusi kelompok | | * Kemampuan dalam menganalisis kompleksitas algoritma greedy * Kemampuan dalam menuliskan definisi formal algoritma greedy | | | | Presentasi  Latihan soal dan implementasi algoritma ke dalam kode program | |  |
| **(13)** | Mahasiswa dapat mengimplementasikan hasil rancangan algoritma dengan menggunakan bahasa OOP dan menguji akurasinya pada suatu online judge | | Evaluasi 2 | | | Ujian tulis, praktek | | * Kemampuan dalam melakukan analisis permasalahan pemrograman dan membuat solusinya menggunakan algoritma greedy * Kemampuan dalam mengimplementasikan algoritma greedy dalam bentuk program | | | | Ujian tulis, praktek | | 20% |
| **(14)** | Mahasiswa mampu menyatakan analisis kebenaran dan kompleksitas secara formal terhadap suatu rancangan algoritma | | String Matching: Naïve, Rabin Karp | | | Ceramah  Diskusi kelompok | | * Kemampuan dalam menganalisis kompleksitas algoritma greedy * Kemampuan dalam menuliskan definisi formal algoritma greedy | | | | Ceramah, latihan soal | |  |
| **(15)** | Mahasiswa mampu menyatakan analisis kebenaran dan kompleksitas secara formal terhadap suatu rancangan algoritma | | String Matching: Knuth Morris Pratt | | | Ceramah  Diskusi kelompok | | * Kemampuan dalam menganalisis kompleksitas algoritma greedy * Kemampuan dalam menuliskan definisi formal algoritma greedy | | | | Ceramah, latihan soal | |  |
| **(16)** | Mahasiswa mampu mempresentasikan keseluruhan tahapan rancangan dan analisis algoritma secara sistematik baik secara lisan maupun tulisan | | Evaluasi Akhir Semester (EAS) | | | Ujian tulis, presentasi, atau praktek | | * Kemampuan dalam melakukan analisis permasalahan pemrograman dan membuat solusinya menggunakan algoritma greedy * Kemampuan dalam mengimplementasikan algoritma greedy dalam bentuk program * Kemampuan dalam mempresentasikan solusi pemrograman yang dibuat. | | | | Ujian tulis, presentasi, praktek | | 30% |

**Catatan :**

1 sks = (50’ TM + 60’ BT + 60’ BM)/Minggu BM = Belajar Mandiri T = Teori (aspek ilmu pengetahuan)

TM = Tatap Muka (Kuliah) PS = Praktikum Simulasi (3 jam/minggu) P = Praktek (aspek ketrampilan kerja)

BT = Belajar Terstruktur. PL = Praktikum Laboratorium (3 jam/minggu)