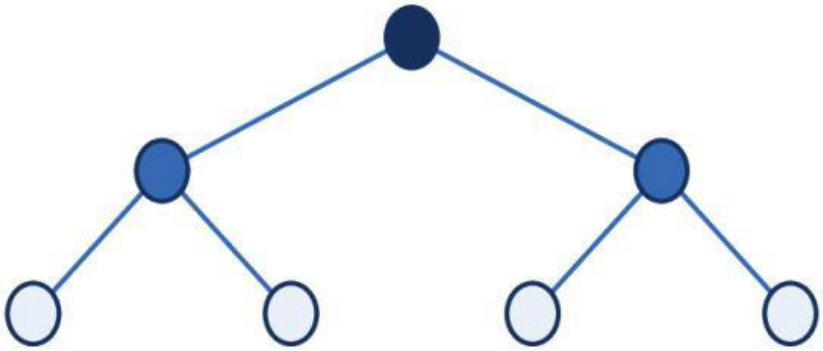
# LEMBAR KERJA PESERTA DIDIK LKPD

INFORMATIKA | SMA KELAS XI | FASE F

## BAB 2

## STRATEGI ALGORITMIK DAN PEMROGRAMAN



Rekursi • Algoritma Greedy • Pemrograman Dinamis
Larik • Karakter dan String

PENULIS

**MAHESWARA PUWA HADI GAUTAMA**

LIVEWORKSHEETS

# A. Identitas

| Identitas Peserta Didik | Isian |
|---|---|
| Nama | _____ ..... ..... |
| Kelas/ Absen | _____ ..... –..... |
| Kelompok | _____ ..... |
| Hari/Tanggal | _____ ..... –..... |
| Guru pengampuh | _____ ..... |

# B. Kata Pengantar

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Esa atas terselesaikannya Lembar Kerja Peserta Didik (LKPD) Informatika Bab 2 ini. LKPD ini disusun sebagai pendamping buku teks agar kalian dapat belajar secara aktif, mandiri, dan bergotong royong dalam memahami strategi algoritmik dan pemrograman.

Program yang kita gunakan sehari-hari, seperti aplikasi belanja, transportasi, dan komunikasi, tidak hanya harus dapat dijalankan, tetapi juga harus benar dan efisien. Untuk itu, kalian perlu menguasai cara menyusun strategi algoritmik, yaitu rekursi, algoritma greedy, dan pemrograman dinamis, serta mampu menuangkannya ke dalam program yang menggunakan larik, karakter, dan string.

LKPD ini memuat tujuan pembelajaran, ringkasan materi, soal pilihan ganda, dan soal uraian beserta pembahasannya. Gunakan LKPD ini dengan sungguh-sungguh, kerjakan soal terlebih dahulu sebelum melihat kunci, dan jangan ragu bertanya kepada guru atau teman jika menemui kesulitan.

Kami menyadari LKPD ini masih jauh dari sempurna. Kritik dan saran yang membangun sangat kami harapkan demi perbaikan di masa mendatang. Selamat belajar dan terus berkarya!

# C. Petunjuk Penggunaan LKPD

1. Tulis identitas kalian pada halaman sampul.
2. Baca tujuan pembelajaran agar mengetahui apa yang harus dicapai.
3. Pelajari ringkasan materi, lalu buka buku teks Bab 2 untuk pendalaman.
4. Kerjakan soal pilihan ganda dan soal uraian secara jujur dan mandiri.
5. Cocokkan jawaban dengan kunci dan pembahasan, lalu catat bagian yang masih belum kalian pahami.

LIVEWORKSHEETS

## D. Tujuan Pembelajaran

**Tujuan umum (sesuai buku):** Setelah mempelajari bab ini, kalian diharapkan mampu menganalisis beberapa strategi algoritmik secara kritis untuk menyelesaikan suatu permasalahan dengan memberikan justifikasi efisiensi, lalu menerapkan strategi tersebut ke dalam program yang lebih kompleks dari program Kelas X. Kalian juga belajar bergotong royong mengembangkan dan menguji program untuk menyelesaikan masalah yang berkaitan dengan mata pelajaran lain.

**Tujuan khusus:** Setelah mengerjakan LKPD ini, peserta didik mampu:

1. Menjelaskan empat tahap proses pemrograman (menganalisis permasalahan, mendesain solusi, mengimplementasikan solusi, dan menguji program) beserta sifatnya yang berulang (siklus).
2. Menyusun definisi rekursif yang terdiri atas basis dan rekursi, serta menggunakannya untuk menyelesaikan permasalahan seperti barisan, faktorial, dan pemindahan panekuk.
3. Menerapkan algoritma greedy pada permasalahan optimasi dan menjelaskan kondisi ketika greedy tidak menghasilkan solusi optimal.
4. Menerapkan pemrograman dinamis dan teknik memoisasi pada permasalahan optimasi yang sub-masalahnya saling beririsan.
5. Membandingkan algoritma greedy dan pemrograman dinamis beserta kelebihan dan kekurangannya.
6. Menggunakan larik (satu dan dua dimensi) serta karakter dan string dalam program.
7. Bekerja sama merancang algoritma dan membuat program untuk permasalahan lintas bidang.

LIVEWORKSHEETS

# E. Ringkasan Materi

**Kata kunci:** strategi algoritmik, pemrograman, penyelesaian masalah.

## A. Proses Pemrograman

Pemrograman adalah seluruh aktivitas teknis untuk menghasilkan suatu program, termasuk analisis kebutuhan serta seluruh langkah desain dan implementasinya. Pemrograman lebih luas daripada sekadar menulis kode (coding). Ada empat langkah utama:

1. **Menganalisis permasalahan (Analyzing).** Pemrogram mendefinisikan masalah dengan berpikir komputasional (abstraksi, dekomposisi, pengenalan pola). Hasilnya adalah pernyataan masalah yang memuat masukan (input), keluaran (output), dan batasan (constraint).
2. **Mendesain solusi (Problem Solving).** Pemrogram merencanakan strategi untuk menghasilkan keluaran dari masukan, boleh memakai solusi yang sudah ada, dan menuliskannya sebagai narasi, pseudocode, atau diagram alir. Algoritma juga dievaluasi, misalnya apakah cukup cepat (kurang dari 1 detik) untuk masukan besar.
3. **Mengimplementasikan solusi (Coding).** Algoritma diubah menjadi kode program dengan bahasa pemrograman tertentu, dengan mempertimbangkan tipe data, struktur kontrol, dan hal teknis lainnya.
4. **Menguji program (Testing).** Program diuji dengan kasus uji (test case) yang mewakili seluruh kemungkinan masukan. Jika ada keluaran yang salah, lakukan analisis ulang dan perbaiki program.

Keempat tahap ini biasanya dilakukan berulang kali sehingga membentuk **siklus pemrograman**. Program yang kompleks dapat dikembangkan secara bertahap (iteratif): tentukan bentuk akhir program, lalu dekomposisi menjadi program-program antara yang lebih sederhana. Biasakan pula membuat **dokumentasi program** (coretan, algoritma, kode sumber, dan hasil pengujian) agar kode mudah dipahami diri sendiri maupun orang lain.

**Contoh kasus: mengidentifikasi jenis segitiga.** Masukan berupa tiga bilangan bulat a, b, c pada rentang [1, 1000]. Pertama, periksa apakah ketiganya dapat membentuk segitiga dengan Teorema Pertidaksamaan Segitiga: $a + b > c$, $a + c > b$, dan $b + c > a$ harus terpenuhi sekaligus. Jika tidak, keluarannya "Bukan Segitiga". Jika ya, tiga sisi sama berarti "Segitiga Sama Sisi", tepat dua sisi sama berarti "Segitiga Sama Kaki", dan ketiga sisi berbeda berarti "Segitiga Sembarang". Program kemudian diuji dengan kasus uji, misalnya 5 5 5, 1000 1000 500, 100 200 500, dan 100 400 200.

## B. Berpikir Komputasional: Strategi Algoritmik

### *1. Rekursi*

Rekursi adalah sesuatu yang mengandung dirinya sendiri (contohnya segitiga Sierpinski dan boneka bersarang Rusia). Suatu masalah dapat didekomposisi menjadi masalah serupa yang ukurannya lebih kecil. Misalnya, menghitung 10! berarti harus menghitung 1! sampai 9! terlebih dahulu.

Definisi barisan atau fungsi rekursif selalu memuat minimal dua hal:

- **Basis:** nilai awal atau dasar, misalnya $a_1 = 1$.

LIVEWORKSHEETS

- **Rekursi:** hubungan nilai saat ini dengan nilai-nilai sebelumnya, misalnya $a_i = a_{i-1} + 2$ untuk $i > 1$.

Banyaknya nilai pada bagian basis ditentukan oleh banyaknya suku sebelumnya yang dipakai pada bagian rekursi. Contohnya **barisan Fibonacci**: $a_1 = a_2 = 1$ dan $a_i = a_{i-1} + a_{i-2}$ untuk $i \geq 2$, menghasilkan 1, 1, 2, 3, 5, 8, 13, 21, dan seterusnya. Karena rekursinya memakai dua suku sebelumnya, basisnya harus mendefinisikan dua nilai awal.

### *2. Algoritma Greedy*

Greedy (rakus) adalah strategi penyelesaian masalah **optimasi**, yaitu mencari hasil terbaik (terkecil atau terbesar), dengan prinsip **mengambil langkah terbaik pada setiap saat**. Dalam konteks informatika kata "greedy" tidak bermakna negatif.

Contoh: Budi memiliki 8 kantong ikan berisi 3, 5, 2, 8, 4, 6, 6, dan 3 ekor, tetapi mobilnya hanya muat 4 kantong. Agar ikan yang terbawa paling banyak, urutkan kantong dari yang terbanyak (8, 6, 6, 5, 4, 3, 3, 2), lalu ambil 4 kantong pertama: 8 + 6 + 6 + 5 = 25 ekor. Langkah **mengurutkan data** terlebih dahulu adalah pola yang umum pada penyelesaian greedy.

**Keterbatasan greedy:** greedy tidak selalu menghasilkan solusi optimal. Pada penukaran uang Rp15.000 dengan pecahan Rp1.000, Rp7.000, dan Rp10.000, greedy memilih Rp10.000 lalu 5 lembar Rp1.000 (total 6 lembar), padahal solusi optimalnya adalah 7.000 + 7.000 + 1.000 (total 3 lembar).

### *3. Pemrograman Dinamis (Dynamic Programming, DP)*

DP cocok untuk permasalahan optimasi yang pilihan terbaik saat ini belum tentu menjadi pilihan terbaik secara keseluruhan. Ada dua unsur utama:

1. **Optimasi melalui serangkaian pilihan.** Semua kombinasi pilihan langkah harus diperhitungkan, karena prinsip greedy belum tentu berlaku.
2. **Nilai optimal dinyatakan secara rekursif dari sub-masalah yang lebih kecil, dan sub-masalah tersebut saling beririsan (overlap).** Agar tidak menghitung hal yang sama berulang-ulang, hasil setiap sub-masalah disimpan dalam tabel. Teknik ini disebut **memoisasi** (dari kata memorandum, "mengingat"), yaitu menyimpan hasil perhitungan agar tidak dihitung dua kali. Memoisasi berbeda dengan memorisasi (menghafal).

Contoh: pada persoalan Agria memanen cabai di kebun berpetak yang hanya boleh bergerak ke kanan atau ke bawah, greedy (mulai dari petak berisi 20 cabai) hanya menghasilkan 26 cabai, sedangkan ada jalur lain dengan 37 cabai. Mencoba semua jalur terlalu banyak dan banyak perhitungan berulang. Dengan DP, isi tabel memoisasi: petak kiri atas diisi nilainya sendiri, sedangkan petak lain diisi **max(A, B) + C**, dengan A nilai tabel di atasnya, B nilai tabel di kirinya (0 jika tidak ada), dan C jumlah cabai pada petak tersebut. Nilai terbesar pada tabel adalah jawabannya (pada contoh buku, 42 cabai).

**Perbandingan singkat.** Greedy sederhana dan cepat, tetapi hanya benar untuk masalah yang pilihan terbaik lokalnya selalu mengarah pada solusi terbaik global. DP lebih umum dan menjamin solusi optimal, tetapi membutuhkan definisi rekursif sub-masalah dan memori untuk tabel memoisasi.

LIVEWORKSHEETS

## C. Algoritma dan Pemrograman

### *1. Larik (Array)*

Untuk mengolah data berukuran besar kita tidak mungkin membuat jutaan variabel tunggal. Solusinya adalah **larik**, yaitu himpunan data yang disimpan dalam satu nama variabel dan diakses menggunakan **indeks**. Analoginya adalah loker bernomor: nomor loker adalah indeks. Saat membuat larik, ukurannya perlu ditentukan. Salah satu implementasinya adalah **vector** dari pustaka C++, yang ukurannya dinamis. Program yang memakai vector disimpan dengan ekstensi .cpp dan dikompilasi dengan kompilator C++ (misalnya GCC).

**Larik dua dimensi** memiliki baris dan kolom seperti matriks, misalnya untuk menyimpan jarak antarkota. Proses ini mencerminkan abstraksi: hubungan antarkota di dunia nyata (kontekstual) disederhanakan menjadi graf (konseptual), lalu menjadi matriks dua dimensi (logikal), yang di memori komputer tersimpan secara linear (fisikal).

### *2. Karakter dan String*

**Karakter** adalah huruf, angka, simbol, atau tanda lain yang mengikuti standar seperti ASCII atau Unicode, dan umumnya bertipe data **char**. **String** adalah rangkaian karakter. Pada bahasa C, string diimplementasikan sebagai larik karakter yang diakhiri karakter '\0'. Bahasa seperti C++ dan Java memiliki tipe data string sendiri. Fungsi pengolah string (mengubah huruf kapital ke non kapital, menggabungkan string, mencari substring, dan sebagainya) tersedia pada pustaka <string.h>. Biasakan membaca dokumentasi bahasa pemrograman dan pustaka yang digunakan.

## D. Latihan Strategi Algoritmik dan Pemrograman Lintas Bidang

Pada bagian ini kalian merancang algoritma, lalu menerjemahkannya ke bahasa pemrograman yang dikuasai (C, Python, dan sebagainya) untuk permasalahan lintas bidang, misalnya simulasi gerak burung yang diluncurkan dengan ketapel (gerak parabola pada Fisika), pengelolaan bank darah, persilangan tanaman, dan stoikiometri. Setiap permasalahan memiliki sub-masalah dengan tingkat kesulitan bertahap, dan kalian bekerja secara gotong royong dalam merancang, membuat, serta menguji program.

## Soal Latihan

### I. Soal Pilihan Ganda

**Petunjuk:** Pilihlah satu jawaban yang paling tepat dengan menulis huruf A, B, C, atau D.

**1.** Urutan langkah pemrograman yang benar adalah ....

A. menguji program, mendesain solusi, menganalisis permasalahan, coding

B. menganalisis permasalahan, mendesain solusi, coding, menguji program

C. coding, menganalisis permasalahan, menguji program, mendesain solusi

D. mendesain solusi, coding, menganalisis permasalahan, menguji program

LIVEWORKSHEETS

**2.** Tahap pemrograman yang menghasilkan pernyataan masalah berisi masukan, keluaran, dan batasan adalah ....
A. menganalisis permasalahan
B. mendesain solusi
C. mengimplementasikan solusi
D. menguji program

**3.** Definisi fungsi atau barisan rekursif minimal harus memuat ....
A. variabel dan konstanta
B. input dan output
C. basis dan rekursi
D. larik dan indeks

**4.** Barisan Fibonacci didefinisikan $a_1 = a_2 = 1$ dan $a_i = a_{i-1} + a_{i-2}$ untuk $i \geq 3$. Nilai $a_7$ adalah ....
A. 8
B. 11
C. 12
D. 13

**5.** Definisi rekursif yang tepat untuk faktorial adalah ....
A. $n! = n + (n - 1)!$ untuk $n > 1$, dan $1! = 1$
B. $n! = n \times (n - 1)!$ untuk $n > 1$, dan $1! = 1$
C. $n! = (n - 1)! - n$ untuk $n > 1$, dan $1! = 1$
D. $n! = n \times n$ untuk $n > 1$, dan $1! = 1$

**6.** Pada permainan memindahkan tumpukan panekuk dari piring A ke piring C (dengan piring bantuan B), memindahkan 2 panekuk membutuhkan minimal 3 langkah. Jumlah langkah minimal untuk 4 panekuk adalah ....
A. 7
B. 8
C. 15
D. 16

**7.** Prinsip utama algoritma greedy adalah ....
A. mengambil langkah terbaik pada setiap saat
B. mencoba semua kemungkinan jalur tanpa terkecuali
C. menyimpan hasil sub-masalah ke dalam tabel
D. membagi data menjadi dua bagian yang sama besar

**8.** Langkah yang umum dilakukan sebelum mengambil keputusan secara greedy adalah ....
A. membuat tabel memoisasi
B. menjumlahkan seluruh data
C. membuat fungsi rekursif
D. mengurutkan data dari terbesar ke terkecil atau sebaliknya

LIVEWORKSHEETS

**9.** Budi memiliki 8 kantong berisi ikan sebanyak 3, 5, 2, 8, 4, 6, 6, dan 3 ekor. Jika mobilnya hanya muat 3 kantong, jumlah ikan terbanyak yang dapat dibawa adalah ... ekor.

A. 18
B. 19
C. 20
D. 21

**10.** Seseorang ingin menukarkan Rp15.000 dengan pecahan Rp1.000, Rp7.000, dan Rp10.000 menggunakan lembar uang sesedikit mungkin. Jumlah lembar minimum adalah ....

A. 2
B. 3
C. 4
D. 6

LIVEWORKSHEETS

## II. Soal Uraian

**Petunjuk:** Jawablah pertanyaan berikut dengan jelas dan runtut.

**1.** Sebutkan dan jelaskan secara singkat empat tahap dalam siklus pemrograman. Mengapa keempat tahap tersebut sering dilakukan berulang kali?

**2.** Sebuah barisan didefinisikan sebagai berikut: $a_1 = 1$, $a_2 = 1$, dan $a_i = a_{i-1} + 2a_{i-2}$ untuk $i \geq 3$. Tentukan nilai suku ke-10 ($a_{10}$).

**3.** Faktorial dari bilangan bulat $n \geq 1$ ditulis $n!$ dan dihitung dengan mengalikan semua bilangan bulat dari 1 sampai n. Buatlah definisi rekursif untuk $n!$, lalu hitung $5!$ dengan menguraikan langkah-langkahnya.

**4.** Sebuah lantai berukuran $2 \times N$ akan dipasangi N buah keramik berukuran $1 \times 2$, yang dapat dipasang mendatar atau tegak. Untuk $N = 4$ terdapat 5 cara pemasangan. Tentukan banyak cara pemasangan untuk $N = 8$ menggunakan konsep rekursi.

**5.** Cici memiliki 10 pekerjaan rumah (PR) dengan waktu pengerjaan (jam): A = 1,5; B = 3; C = 1; D = 0,5; E = 4; F = 1; G = 2,5; H = 1; I = 0,5; J = 2. Waktu yang tersedia hanya 8 jam dan setiap PR bernilai sama. Dengan strategi greedy, PR mana saja yang dikerjakan agar jumlah PR yang selesai sebanyak mungkin, dan berapa jumlahnya?

LIVEWORKSHEETS

# F. REFLEKSI PEMBELAJARAN

*Bab 2: Strategi Algoritmik dan Pemrograman*

Refleksi membantu kalian melihat sejauh mana pemahaman yang sudah dicapai dan apa yang masih perlu diperbaiki. Jawablah dengan jujur sesuai pengalaman kalian selama mempelajari bab ini. Tidak ada jawaban benar atau salah pada lembar ini.

## A. Penilaian Diri terhadap Tujuan Pembelajaran

Berilah tanda centang (✓) pada kolom yang sesuai dengan kemampuan kalian saat ini.

| No. | Saya mampu ... | Paham | Cukup | Belum |
|---|---|---|---|---|
| 1 | menjelaskan empat tahap proses pemrograman dan sifatnya yang berulang (siklus) | | | |
| 2 | menyusun definisi rekursif (basis dan rekursi) untuk barisan, faktorial, dan masalah sejenis | | | |
| 3 | menerapkan algoritma greedy pada masalah optimasi | | | |
| 4 | menjelaskan kapan greedy tidak menghasilkan solusi optimal | | | |
| 5 | menerapkan pemrograman dinamis dan memoisasi untuk submasalah yang saling beririsan | | | |
| 6 | membandingkan algoritma greedy dan pemrograman dinamis | | | |

LIVEWORKSHEETS

| | | | | |
|---|---|---|---|---|
| 7 | menggunakan larik (satu dan dua dimensi) dalam program | | | |
| 8 | menggunakan karakter dan string dalam program | | | |
| 9 | bekerja sama merancang algoritma dan menguji program untuk masalah lintas bidang | | | |

## B. Pertanyaan Refleksi

1. Konsep apa yang paling kalian pahami pada Bab 2? Jelaskan dengan kata-kata sendiri.

2. Bagian materi mana (proses pemrograman, rekursi, greedy, pemrograman dinamis, larik, ataustring) yang menurut kalian paling sulit? Mengapa?

3. Menurut kalian, apa perbedaan penting antara algoritma greedy dan pemrograman dinamis? Berikan satu contoh masalah untuk masing-masing.

LIVEWORKSHEETS

4. Bagaimana materi pada bab ini dapat kalian gunakan dalam kehidupan sehari-hari atau padamata pelajaran lain?

5. Bagaimana kerja sama dengan teman atau kelompok membantu kalian memahami materi danmemeriksa kebenaran program?

6. Langkah apa yang akan kalian lakukan untuk memperbaiki bagian yang masih belumdipahami?

LIVEWORKSHEETS

## C. Kesimpulan Pembelajaran

Lengkapilah kalimat berikut.

Setelah mempelajari bab ini, saya dapat ...

Hal yang paling berkesan bagi saya adalah ...

Saya masih ingin belajar lebih lanjut tentang ...

## D. Tingkat Keyakinan Diri

Lingkari angka yang menggambarkan keyakinan kalian dalam menerapkan strategi algoritmik pada masalah baru (1 = belum yakin, 5 = sangat yakin).

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

**Mengetahui,**
**Guru**
**Pengampu**
**(.....)**

**.....,**
**..... 20....**
**Peserta Didik**
**(.....)**



LIVEWORKSHEETS