أولمبياد البرمجة

البدء في البرمجة التنافسية

جهّز بيئة عملك، تعلّم C++ بالطريقة التي تستخدمها المسابقات فعليًا، واحصل على أول حل مقبول لديك على حكم برمجي حقيقي.

المستوى: مبتدئ **مدة التنفيذ:** عطلة نهاية أسبوع واحدة حتى أول تسليم **التكلفة التقريبية:** $0

قبل البدء: لست بحاجة لمعرفة C++ مسبقًا — منطق أساسي بأي لغة (Python أو Java أو Scratch) يكفي. ما يهم هو الراحة مع المتغيرات والحلقات والدوال.

01 الهدف

تُعرف البرمجة التنافسية بمنحنى دخول شديد الانحدار — ليس لأن المسائل الأولى صعبة، بل لأن المبتدئين عادة يقفزون مباشرة إلى مسائل صعبة دون بيئة عمل جاهزة، أو قالب كود، أو أي فهم لما يعنيه "مقبول" مقابل "إجابة خاطئة". بنهاية هذا المخطط، ستملك إعداد C++ عاملًا، وستكون قد أرسلت وحللت أول مسائلك على حكم برمجي حقيقي عبر الإنترنت، وستفهم خريطة طريق مواضيع الخوارزميات أمامك — بحيث تذهب ساعاتك المئة القادمة من التدريب إلى مكان ما، بدلًا من الدوران في حلقة مفرغة.

02 المتطلبات الأساسية

- إلمام برمجي أساسي بأي لغة — المتغيرات، الحلقات، الشروط، الدوال
- حاسوب قادر على تثبيت مترجم (يعمل على Windows أو macOS أو Linux)
- الراحة في كتابة الكود دون إكمال تلقائي كثيف — المسابقات تكافئ الكتابة السريعة والصحيحة للكود

03 الأدوات الموصى بها

الأداة	الغرض	ملاحظات
g++ (MinGW على Windows، أو مدمج في macOS/Linux)	مترجم C++	مجاني — C++ هي اللغة السائدة في المسابقات بسبب سرعتها
VS Code	محرر الأكواد	مجاني، مع تثبيت إضافة C/C++
حساب Codeforces	الحكم الرئيسي للتدريب والمسابقات	مجاني — أكبر مجتمع نشط للبرمجة التنافسية
حساب AtCoder	مسابقات مناسبة للمبتدئين	مجاني — معروف بمسائل جيدة الصياغة وتدرّج صعوبة واضح
Competitive Companion (إضافة متصفح)	يحلّل المسائل تلقائيًا إلى محررك	مجاني، اختياري لكنه يوفر وقت الإعداد
مستودع GitHub شخصي للمسائل المحلولة	يتتبع التقدم، يصبح عملًا يُعرض في ملف الإنجازات	مجاني

04 الأدوات المطلوبة

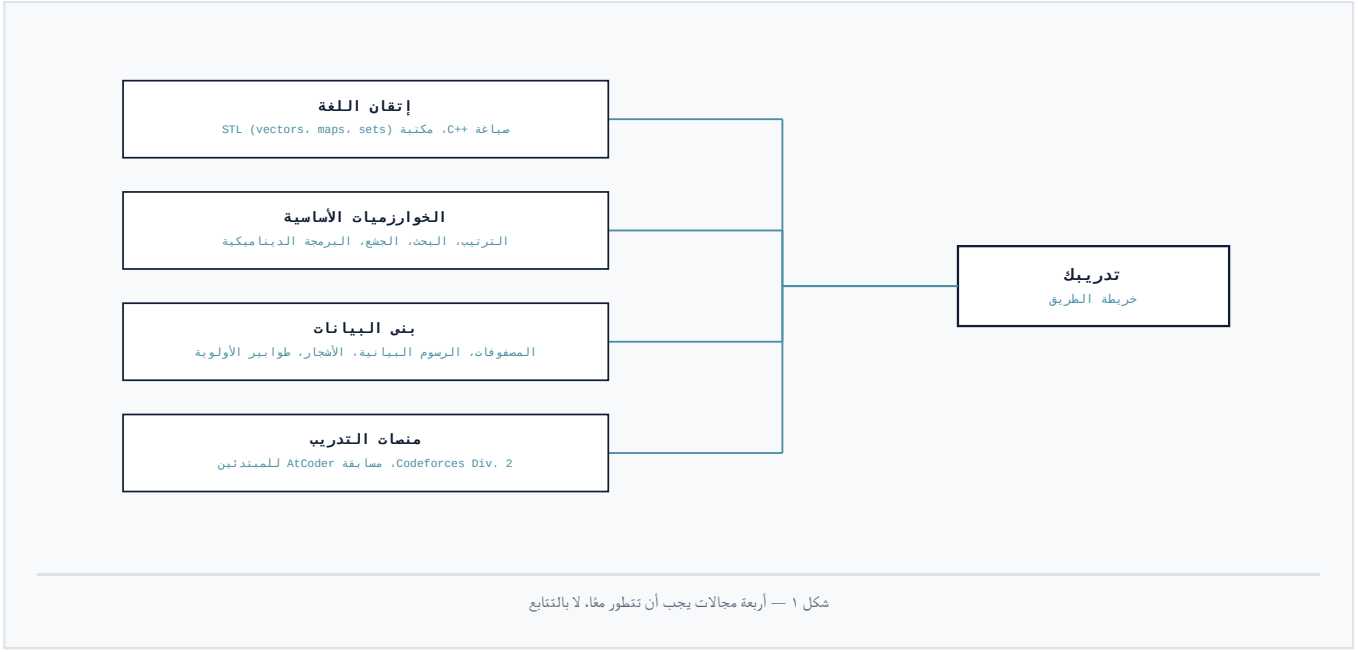
- حاسوب مثبّت عليه g++ و VS Code
- حساب مجاني على Codeforces (codeforces.com)

نزاهة المسابقات

لا تستخدم أبدًا حسابًا ثانيًا لحصد مسائل أسهل من أجل التصنيف، ولا تقدّم أبدًا حلًا نسخته دون فهمه — كلاهما يُعامَل كغش على كل حكم برمجي رئيسي وقد يؤدي إلى حظر دائم للحساب. القيمة الكاملة لهذا التدريب هي ما يحدث في عقلك، لا رقم التصنيف.

05 مسار تعلّمك

البرمجة التنافسية ليست مهارة واحدة — بل إتقان اللغة، والمعرفة الخوارزمية، ومعرفة بنى البيانات تتطور معًا. تقدّم في الثلاثة جميعًا، لا في الأكثر متعة فقط.



شكل ١ — أربعة مجالات يجب أن تتطور معًا، لا بالتتابع

المجال	نقطة البداية	الهدف
اللغة	أساسيات C++ وvector وmap وset من STL	كتابة كود صحيح بسرعة، دون البحث عن الصياغة
الخوارزميات	الترتيب، البحث الثنائي، القوة الغاشمة	التعرف على التقنية التي تطلبها المسألة
بنى البيانات	المصفوفات والرسوم البيانية البسيطة	اختيار البنية الصحيحة قبل الكتابة، لا في منتصف الطريق
التدريب	مسائل Codeforces Div. 2 من فئة A/B	الحل باستمرار، ثم رفع الصعوبة تدريجيًا

06 خطوات البناء

1. **ثبّت g++ وتأكد من عمله.** ترجم وشغّل "Hello World" من الطرفية قبل لمس أي موقع مسابقات.
2. **اكتب ملف قالب قابل لإعادة الاستخدام.** إعداد إدخال/إخراج سريع، الاستدعاءات الشائعة، وهيكل `main()` تنسخه لكل مسألة — انظر القسم ٠٧.
3. **أنشئ حسابات الحكم البرمجي.** Codeforces و AtCoder كلاهما مجاني ومستخدم على نطاق واسع.
4. **حل أول ١٠ مسائل عند أسهل مستوى تصنيف.** على Codeforces، صفِّ حسب تصنيف ≥ ٨٠٠. الهدف هنا بناء عادة الإرسال-التصحيح-إعادة الإرسال، لا الصعوبة.
5. **تعلّم تحليل التعقيد الأساسي.** افهم لماذا يفشل حل $O(n^2)$ زمنيًا عند $n = 10^6$ — هذا المفهوم الوحيد يمنع معظم أحكام "تجاوز الوقت المحدد" المبكرة.
6. **ضع حصة أسبوعية من المسائل.** ٣–٥ مسائل أسبوعيًا، باستمرار، أفضل من ٢٠ مسألة دفعة واحدة تليها شهر توقف.
7. **خض أول مسابقة لك — افتراضيًا، لا مباشرة.** تتيح لك Codeforces خوض المسابقات السابقة "افتراضيًا" تحت ضغط زمني حقيقي دون قلق لوحة الترتيب المباشرة.
8. **حل ما تبقى بعد كل مسابقة.** ارجع وحل المسائل التي لم تنهها، مستخدمًا الشرح الرسمي فقط بعد محاولة حقيقية — هنا يحدث معظم التعلم الفعلي.

07 شرح الكود

هذا قالب قياسي للبرمجة التنافسية — إدخال/إخراج سريع، استدعاءات شائعة، وبنية نظيفة ستعيد استخدامها لمئات المسائل. تحته، أول مسألة محلولة توضّح النمط عمليًا.

```
#include <bits/stdc++.h>
using namespace std;

// Fast I/O — saves time on problems with large input
static void fastio() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
}

int main() {
    fastio();

    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
```

```
    // Example: find two indices whose values sum to a target
    int target;
    cin >> target;
    unordered_map<int, int> seen;
    for (int i = 0; i < n; i++) {
        int need = target - a[i];
        if (seen.count(need)) {
            cout << seen[need] << " " << i << "\n";
            return 0;
        }
        seen[a[i]] = i;
    }
    cout << -1 << "\n";
    return 0;
}
```

يعمل هذا في زمن $O(n)$ باستخدام جدول تجزئة بدلًا من طريقة الحلقة المتداخلة ($O(n^2)$ — وهو الفرق بين النجاح وتجاوز الوقت المحدد على مدخلات كبيرة.

08 الاختبار واستكشاف الأخطاء

الحكم / العارض	السبب المحتمل	الحل
تجاوز الوقت المحدد (TLE)	تعقيد الخوارزمية مرتفع جدًا لحجم المدخل	تحقق من القيود المذكورة في المسألة؛ حلقة $O(n^2)$ تفشل عندما يتجاوز n نحو 10^4–10^5
إجابة خاطئة في الاختبار ٢ أو ٣	لم تُعالَج حالة طرفية (n=0، أعداد سالبة، تكرارات)	اختبر يدويًا مدخلات صغيرة ومتطرفة قبل الإرسال
خطأ ترجمة على الحكم فقط، وليس محليًا	الحكم يستخدم معيار C++ أحدث أو أقدم وأكثر صرامة	طابق إعدادات مترجمك المحلي مع إعدادات الحكم (عادة C++17)
انخفاض التصنيف ويشعرك بالإحباط	مقارنة نمو تصنيفك بالآخرين بدلًا من مستواك الأساسي الخاص	تتبع المسائل المحلولة والمفاهيم المتعلَّمة، لا التصنيف فقط — التصنيف متذبذب على المدى القصير
العلوق لأكثر من ساعة في مسألة مستوى مبتدئ	غياب نمط محدد، لا فجوة مهارة عامة	اقرأ الشرح الرسمي بعد محاولة حقيقية — هذا طبيعي، وليس إشارة للتوقف

09 الخطوة التالية

من مسائل المسابقات إلى مجموعة مهارات موثّقة

ملف تعريف على Codeforces وحده ليس ملف إنجازات — مستودع GitHub بمسائلك المحلولة منظمة حسب الموضوع، مع ملاحظات موجزة عن التقنية التي علّمتك إياها كل مسألة، هو ما يُعتبر كذلك. اجمع هذا المخطط مع **SA-001 (بناء أول مشروع حقيقي بالذكاء الاصطناعي)** لتحويل مهارتك الخوارزمية إلى أداة قابلة للنشر، أو استمر مع **PO-002 (بنى البيانات لبرمجة المسابقات)** عندما تصبح المسائل الأساسية روتينية. راجع وحدة دليل الجامعات والمنح لمعرفة كيف تُقرأ نتائج البرمجة التنافسية من قبل لجان القبول.