प्रतियोगी प्रोग्रामिंग

1. कम से कम एक भाषा को पूरी तरह से सीखें, विशेष रूप से □++ के लिए गति और नियंत्रण के लिए.
2. भाषा-विशिष्ट अनुकूलन, जैसे □++ में तेज □/□ को समझें.
3. मानक पुस्तकालयों और उनके कार्यों से परिचित हों.
4. डेटा को दक्षतापूर्वक संग्रहित और पहुंचने के लिए, एरेस फंडामेंटल हैं।
5. लिंक्ड लिस्ट डायनामिक डेटा संग्रहण के लिए उपयोगी हैं।
6. स्टैक और क्यू □□□□ और □□□□ ऑपरेशंस को क्रमशः लागू करते हैं।
7. हैश टेबल □(1) औसत मामले में खोज और इनसर्ट करने के लिए प्रदान करते हैं।
8. पेड़, विशेष रूप से बाइनरी पेड़ और बाइनरी सर्च पेड़, हायरार्किकल डेटा के लिए आवश्यक हैं।
9. ग्राफ संबंधों को मॉडल करते हैं और कई एल्गोरिथम के केंद्र में हैं।
10. हिप्स प्रायोरिटी क्यू इम्प्लिमेंटेशन के लिए उपयोग किए जाते हैं।
11. सेगमेंट पेड़ और फेनविक पेड़ (□□□) रेंज क्वेरी और अपडेट के लिए महत्वपूर्ण हैं।

एल्गोरिथम सेक्शन:

12. क्विकसॉर्ट और मर्जसॉर्ट जैसे सॉर्टिंग एल्गोरिथम फंडामेंटल हैं।
13. बाइनरी सर्च सॉर्टेड डेटा में लॉगारिथ्मिक खोज के लिए आवश्यक है।
14. डायनमिक प्रोग्रामिंग समस्याओं को उपसमस्याओं में तोड़कर हल करता है।
15. □□□ और □□□ ग्राफ ट्रेवर्सल के लिए उपयोग किए जाते हैं।
16. डिज्कस्ट्रा एल्गोरिथम नॉन-नैगेटिव वेट्स वाले ग्राफ में सबसे छोटी पथ खोजता है।
17. क्रुस्कल और प्रिम एल्गोरिथम ग्राफ का मिनिमम स्पैनिंग ट्री खोजते हैं।
18. ग्रीडी एल्गोरिथम प्रत्येक चरण में स्थानीय रूप से सर्वोत्तम चुनाव करते हैं।
19. बैकट्रैकिंग एक्सपोनेंशियल समय जटिलता वाले समस्याओं, जैसे □-क्वीन के लिए उपयोग किया जाता है।
20. संख्या सिद्धांत के अवधारणाएं जैसे □□□, □□□, प्राइम फैक्टोराइजेशन अक्सर उपयोग में आती हैं।
21. संयोजनिकी गणना समस्याओं, अनुक्रमण और संयोजन के लिए।
22. प्रायिकता और अपेक्षित मान में अकस्मात समस्याओं में शामिल हैं।
23. ज्यामिति समस्याएं बिंदुओं, रेखाओं, बहुभुजों और वृत्तों से संबंधित होती हैं।
24. बिग ओ नोटेशन के लिए समय और स्थान जटिलता समझें।
25. मेमोइजेशन को महंगे फंक्शन कॉल के परिणामों को स्टोर करने के लिए उपयोग करें।

26. लूप्स को ऑप्टिमाइज करें और अनावश्यक गणनाओं से बचें।

27. बिट मैनिपुलेशन बाइनरी डेटा पर दक्ष ऑपरेशंस के लिए उपयोग करें।

28. डिवाइड एंड कंकर समस्याओं को छोटे, प्रबंधनीय उपसमस्याओं में तोड़ता है।

29. दो-पॉइंटर तकनीक सॉर्टेड एरेस और जोड़ों को खोजने के लिए उपयोगी है।

30. स्लाइडिंग विंडो समस्याओं में सबएरेस या सबस्ट्रिंग शामिल हैं।

31. बिटमास्किंग उपसमूहों को दर्शाता है और स्टेट प्रतिनिधित्व में उपयोगी है।

32. कोडफोर्सेस एक व्यापक समस्या सेट और नियमित प्रतियोगिताओं का समर्थन करता है।

33. लीटकोड इंटरव्यू-शैली समस्याओं के लिए अच्छा है।

34. हैकररैंक विभिन्न चुनौतियों और प्रतियोगिताओं का समर्थन करता है।

35. रेटिंग प्रणाली और समस्या कठिनाई स्तरों को समझें।

36. टाइम्ड परिस्थितियों में अभ्यास करें ताकि प्रतियोगिता वातावरण को सिमुलेट करें।

37. समय को प्रभावी ढंग से प्रबंधित करें, आसान समस्याओं से शुरू करें।

38. □□□/□□□□ में टीम सहयोग के लिए एक रणनीति विकसित करें।

39. □□□ समस्याएं एल्गोरिथमिक होती हैं और अक्सर गहन समझ की आवश्यकता होती है।

40. □□□/□□□□ टीमवर्क और तेज समस्या-समाधान पर जोर देता है।

41. "इंट्रॉडक्शन टू एल्गोरिथम्स"सीएलआरएस जैसी किताबें आवश्यक हैं।

42. कोर्सेरा और □□□ जैसे प्लेटफॉर्मों पर ऑनलाइन कोर्स।

43. ट्यूटोरियल और व्याख्याओं के लिए यूट्यूब चैनल।

44. चर्चा के लिए फोरम और समुदाय में भाग लेना।

45. यूनियन-फाइंड (डिसजॉइंट सेट यूनियन) कनेक्टिविटी समस्याओं के लिए।

46. □□□ अनवैटेड ग्राफ में सबसे छोटी पथ के लिए।

47. □□□ ग्राफ ट्रेवर्सल और टोपोलॉजिकल सॉर्टिंग के लिए।

48. क्रुस्की एल्गोरिथम यूनियन-फाइंड को □□□ के लिए उपयोग करता है।

49. प्रिम एल्गोरिथम एक शुरुआती वर्टेक्स से □□□ बनाता है।

50. बेलमैन-फोर्ड ग्राफ में नैगेटिव साइकिल्स का पता लगाता है।

51. फ्लॉयड-वार्शॉल सभी-पेयर शॉर्टेस्ट पथ को कंप्यूट करता है।

52. बाइनरी सर्च एक्सपोनेंशियल फंक्शंस में शामिल समस्याओं में भी उपयोग किया जाता है।

53. प्रिफिक्स समस्या के लिए रेंज क्वेरी ऑप्टिमाइजेशन।

54. एरटोथेनेसियन सिव के लिए प्राइम नंबर जनरेशन।

55. एडवांस्ड पेड़ जैसे □□□ और रेड-ब्लैक पेड़ संतुलन बनाए रखते हैं।

56. ट्राई के लिए दक्ष प्रिफिक्स खोज।

57. सेगमेंट पेड़ रेंज क्वेरी और अपडेट को दक्षतापूर्वक समर्थन करते हैं।

58. फेनविक पेड़ सेगमेंट पेड़ से आसान हैं।

59. स्टैक के लिए एक्सप्रेशन पर्सिंग और पैरेंथेसिस बैलेंसिंग।

60. क्यू के लिए □□□ और अन्य □□□□ ऑपरेशंस।

61. डीक्यू के लिए दोनों छोरों से दक्ष इनसर्ट और डिलीट।

62. हैशमैप के लिए फास्ट एक्सेस के साथ की-वैल्यू स्टोरेज।

63. ट्रीसेट के लिए ऑर्डर्ड की स्टोरेज के साथ लॉग एन ऑपरेशंस।

64. मोड्यूलर अरिथमेटिक बड़े संख्याओं से संबंधित समस्याओं के लिए महत्वपूर्ण है।

65. तेज गणना के लिए पावर्स को कंप्यूट करने के लिए फास्ट एक्स्पोनेंशन।

66. लिनियर रिकर्सेस को हल करने के लिए मैट्रिक्स एक्स्पोनेंशन।

67. यूक्लिड एल्गोरिथम के लिए □□□ कंप्यूटेशन।

68. संयोजनिकी में इनक्लूजन-एक्सक्लूजन सिद्धांत।

69. प्रायिकता वितरण और अपेक्षित मान सिमुलेशन में।

70. प्लेन ज्यामिति अवधारणाएं जैसे बहुभुजों का क्षेत्र और कॉन्वेक्स हल।

71. रेखा इंटरसेक्शन जैसे कंप्यूटेशनल ज्यामिति एल्गोरिथम।

72. जब संभव हो, रिकर्सन से बचें और इटरेटिव सॉल्यूशन का उपयोग करें।

73. कुछ परिस्थितियों में गति के लिए बिटवाइज ऑपरेशंस का उपयोग करें।

74. जब संभव हो, गणना समय बचाने के लिए मान्यताओं को प्रीकंप्यूट करें।

75. मेमोइजेशन को दक्षतापूर्वक उपयोग करें ताकि स्टैक ओवरफ्लो से बचा जा सके।

76. ग्रीडी एल्गोरिथम अक्सर शेड्यूलिंग और संसाधन आवंटन में उपयोग किए जाते हैं।

77. डायनमिक प्रोग्रामिंग ऑप्टिमाइजेशन समस्याओं के लिए शक्तिशाली है।

78. स्लाइडिंग विंडो को कुछ गुणों वाले सबएरेस खोजने के लिए लागू किया जा सकता है।

79. बैकट्रैकिंग एक्सपोनेंशियल खोज स्पेस वाले समस्याओं के लिए आवश्यक है।

80. डिवाइड एंड कंकर सॉर्टिंग और खोज एल्गोरिथम के लिए उपयोगी है।

81. कोडफोर्सेस एक रेटिंग प्रणाली है जो समस्या कठिनाई को दर्शाता है।

82. वास्तविक प्रतियोगिता अनुभव को सिमुलेट करने के लिए वर्चुअल प्रतियोगिताओं में भाग लेना।
83. कोडफोर्सेस के समस्या टैग्स को उपयोग करें ताकि विशेष विषयों पर ध्यान केंद्रित किया जा सके।
84. लीटकोड इंटरव्यू प्रश्नों और सिस्टम डिजाइन समस्याओं पर ध्यान केंद्रित करता है।
85. हैकररैंक विभिन्न चुनौतियों का समर्थन करता है, जिसमें □□ और मशीन लर्निंग शामिल हैं।
86. प्रतियोगिता अनुभव प्राप्त करने के लिए पिछले प्रतियोगिताओं में भाग लेना।
87. समस्याओं के समाधानों को समीक्षित करने के लिए प्रतियोगिता के बाद नए तकनीकों सीखें।
88. कमजोर क्षेत्रों में समस्याओं का अभ्यास करके सुधार करें।
89. एक समस्या नोटबुक का उपयोग करें ताकि महत्वपूर्ण समस्याएं और समाधानों का पालन किया जा सके।
90. □□□ समस्याएं अक्सर जटिल एल्गोरिथम और डेटा स्ट्रक्चर्स की आवश्यकता होती हैं।
91. □□□/□□□□ तेज कोडिंग और प्रभावी टीम समन्वय की आवश्यकता होती है।
92. प्रत्येक प्रतियोगिता के नियम और फॉर्मेट को समझें ताकि अनुकूल हो सके।
93. “द कंप्यूटर प्रोग्रामिंग आर्ट”क्नुथ एक क्लासिक संदर्भ है।
94. “एल्गोरिथम डिजाइन”क्लाइनबर्ग और टार्डोस एडवांस्ड टॉपिक्स को कवर करता है।
95. “कम्पेटिटिव प्रोग्रामिंग 3”स्टीवन और फेलिक्स हलिम एक गेटो बुक है।
96. ऑनलाइन जज जैसे □□□□, कोडचेफ और एटकोडर विविध समस्याएं प्रदान करते हैं।
97. टिप्स और व्याख्याओं के लिए प्रतियोगी प्रोग्रामिंग ब्लॉग और यूट्यूब चैनल फॉलो करें।
98. कोडिंग समुदाय जैसे स्टैक ओवरफ्लो और रेडिट में भाग लेना।
99. पैटर्न खोज के लिए क्नुथ-मोरिस-प्रैट (□□□) एल्गोरिथम।
100. पैटर्न मैचिंग के लिए □-एल्गोरथिम।
101. बहु-पैटर्न खोज के लिए अहो-कोरासिक।
102. मैक्सिमम फ्लो एल्गोरिथम जैसे फोर्ड-फुल्कर्सन और डिनिक्स एल्गोरिथम।
103. मिनिमम कट और बिपार्टाइट मैचिंग समस्याएं।
104. दक्ष स्ट्रिंग तुलना के लिए स्ट्रिंग हैशिंग।
105. स्ट्रिंग तुलना के लिए लॉंगेस्ट कॉमन सबसीक्वेंस (□□□)।
106. स्ट्रिंग ट्रांसफॉर्मेशन के लिए एडिट डिस्टेंस।
107. पेलिंड्रोमिक सबस्ट्रिंग खोजने के लिए मैनाचर्स एल्गोरिथम।
108. एडवांस्ड स्ट्रिंग प्रोसेसिंग के लिए सफिक्स एरेस।
109. डायनामिक सेट्स के लिए संतुलित बाइनरी सर्च पेड़।

110. ट्रेप्स पेड़ और हिप्स को संयोजित करने के लिए दक्ष ऑपरेशंस।

111. यूनियन-फाइंड के साथ पाथ कम्प्रेशन और यूनियन बाय रैंक।

112. रेंज मिनिमम क्वेरी के लिए स्पार्स टेबल।

113. डायनामिक ग्राफ समस्याओं के लिए लिंक-कट पेड़।

114. ग्राफ में कनेक्टिविटी के लिए डिसजॉइंट सेट्स।

115. सिमुलेशन में घटनाओं को प्रबंधित करने के लिए प्रायोरिटी क्यू।

116. प्रायोरिटी क्यू इम्प्लिमेंटेशन के लिए हिप्स।

117. ग्राफ एडजेन्सी लिस्ट्स □□. एडजेन्सी मैट्रिक्स।

118. ट्री ट्रेवर्सल के लिए यूलर टूर।

119. यूलर टोटिएंट फंक्शन जैसे संख्या सिद्धांत अवधारणाएं।

120. मोड्यूलर इनवर्स के लिए फर्माट्स लिटिल थीअरम।

121. चाइनीज रिमेंडर थीअरम के लिए सिस्टम ऑफ कॉनग्रूएन्स को हल करने के लिए।

122. लिनियर ट्रांसफॉर्मेशन के लिए मैट्रिक्स मल्टीप्लिकेशन।

123. पॉलिनोमियल मल्टीप्लिकेशन के लिए फास्ट फूरियर ट्रांसफॉर्म (□□□)।

124. मार्कोव चेन और स्टोकास्टिक प्रोसेस में प्रायिकता।

125. रेखा इंटरसेक्शन और कॉन्वेक्स हल जैसे ज्यामिति अवधारणाएं।

126. कंप्यूटेशनल ज्यामिति समस्याओं के लिए प्लेन स्विप एल्गोरिथम।

127. दक्ष बूलियन ऑपरेशंस के लिए बिटसेट का उपयोग करें।

128. बुल्क में पढ़ने के लिए □/□ ऑपरेशंस को ऑप्टिमाइज करें।

129. प्रिसिजन त्रुटियों से बचने के लिए फ्लोटिंग पॉइंट्स का उपयोग करने से बचें।

130. जब संभव हो, ज्यामिति गणनाओं के लिए इंटीजर अरिथमेटिक का उपयोग करें।

131. संयोजनिकी के लिए फैक्टोरियल्स और इनवर्स फैक्टोरियल्स को प्रीकंप्यूट करें।

132. मेमोइजेशन और □□ टेबल को दक्षतापूर्वक उपयोग करें ताकि स्थान बचा जा सके।

133. समस्याओं को ज्ञात एल्गोरिथमिक समस्याओं में घटाएं।

134. जटिल समस्याओं को सरल बनाने के लिए अवधारणाओं का उपयोग करें।

135. एज केस और सीमा स्थितियों को सावधानी से विचार करें।

136. स्थानीय रूप से निर्धारित सर्वोत्तम चुनावों के लिए ग्रीडी एप्रोच का उपयोग करें।

137. ओवरलैपिंग सबप्रॉब्लम्स और ऑप्टिमल सबस्ट्रक्चर वाले समस्याओं के लिए □□ का उपयोग करें।

138. सभी संभव समाधानों को खोजने के लिए बैकट्रैकिंग का उपयोग करें।

139. कोडफोर्सेस विशेष विषयों पर ध्यान केंद्रित करने वाले शैक्षिक राउंड्स का समर्थन करता है।

140. लीटकोड बाइवीकली प्रतियोगिताएं और समस्या सेट प्रदान करता है।

141. हैकररैंक एल्गोरिथम, डेटा स्ट्रक्चर्स और गणित जैसे डोमेन-विशिष्ट चुनौतियां प्रदान करता है।

142. विश्व स्तर के प्रतियोगिताओं में भाग लेना ताकि सर्वश्रेष्ठ प्रोग्रामरों के साथ प्रतिस्पर्धा की जा सके।

143. समस्या फिल्टर का उपयोग करें ताकि विशेष कठिनाई और विषयों पर अभ्यास किया जा सके।

144. समस्या रैंकिंग का विश्लेषण करें ताकि कठिनाई का अनुमान लगाया जा सके और सुधार क्षेत्रों पर ध्यान केंद्रित किया जा सके।

145. प्रतियोगिता के दौरान एक व्यक्तिगत समस्या-समाधान रणनीति विकसित करें और उस पर अडिग रहें।

146. समय दबाव में कोडिंग अभ्यास करें ताकि गति और सटीकता में सुधार हो सके।

147. प्रतियोगिता के दौरान कोड को दक्षतापूर्वक समीक्षित और डिबग करें।

148. सहीता की पुष्टि करने के लिए टेस्ट केस का उपयोग करें।

149. उच्च दबाव वाले परिस्थितियों में तनाव को प्रबंधित करने और ध्यान केंद्रित रखने के लिए सीखें।

150. □□□/□□□□ में टीम सदस्यों के साथ प्रभावी ढंग से सहयोग करें।

151. □□□ समस्याएं अक्सर गहन एल्गोरिथमिक अवधारणाओं और दक्षतापूर्वक इम्प्लिमेंटेशन की आवश्यकता होती हैं।

152. □□□/□□□□ टीमवर्क, संचार और तेज फैसला लेने पर जोर देता है।

153. विभिन्न प्रतियोगिताओं में स्कोरिंग और पेनल्टी प्रणाली को समझें।

154. □□□ और □□□/□□□□ के पिछले समस्याओं का अभ्यास करें ताकि शैली से परिचित हो सकें।

155. ट्यूटोरियल और व्याख्याओं के लिए प्रतियोगी प्रोग्रामिंग यूट्यूब चैनल फॉलो करें।

156. समस्याओं और समाधानों के बारे में चर्चा करने के लिए ऑनलाइन समुदाय और फोरम में शामिल हों।

157. ऑनलाइन जज का उपयोग करें ताकि समस्याओं का अभ्यास किया जा सके और प्रगति का पालन किया जा सके।

158. गहन सीखने के लिए वर्कशॉप, सेमिनार और कोडिंग कैम्प में भाग लेना।

159. समस्या हल करने के बाद एडिटोरियल और समाधान पढ़ें ताकि विकल्पों के बारे में सीखें।

160. नवीनतम एल्गोरिथम और तकनीकों के बारे में अनुसंधान पत्रों और लेखों के माध्यम से अपडेट रहें।

161. ऑप्टिमाइजेशन समस्याओं के लिए लिनियर प्रोग्रामिंग।

162. संसाधन आवंटन के लिए नेटवर्क फ्लो एल्गोरिथम।

163. पैटर्न मैचिंग और मैनिपुलेशन के लिए स्ट्रिंग एल्गोरिथम।

164. टार्जन के स्ट्रांगली कनेक्टेड कम्पोनेंट जैसे एडवांस्ड ग्राफ एल्गोरिथम।

165. ट्री समस्याओं के लिए सेंट्रॉइड डिकम्पोजिशन।

166. दक्ष ट्री क्वेरी के लिए हेवी-लाइट डिकम्पोजिशन।

167. डायनामिक ग्राफ कनेक्टिविटी के लिए लिंक-कट पेड़।

168. लेजी प्रोपेगेशन के साथ सेगमेंट पेड़ रेंज अपडेट के लिए।

169. प्रिफिक्स समस्या और अपडेट के लिए बाइनरी इंडेक्सेड पेड़।

170. दक्ष प्रिफिक्स खोज और ऑटोकम्प्लीट फीचर्स के लिए ट्राई।

171. फाइबोनाची हिप्स जैसे एडवांस्ड हिप्स इम्प्लिमेंटेशन।

172. यूनियन-फाइंड के साथ यूनियन बाय रैंक और पाथ कम्प्रेशन।

173. दक्ष स्ट्रिंग प्रोसेसिंग के लिए सफिक्स ऑटोमेटा।

174. डायनामिक ग्राफ ऑपरेशंस के लिए लिंक-कट पेड़।

175. वर्जनिंग और ऐतिहासिक डेटा एक्सेस के लिए पर्सिस्टेंट डेटा स्ट्रक्चर्स।

176. दक्ष स्ट्रिंग मैनिपुलेशन के लिए रोप डेटा स्ट्रक्चर्स।

177. तेज ऑपरेशंस के लिए वैन एमडे बोआस पेड़।

178. चेनिंग और ओपन एड्रेसिंग के साथ हैश टेबल।

179. प्रोबेबलिस्टिक सेट मेम्बरशिप के लिए ब्लूम फिल्टर।

180. कम्पैक्ट स्ट्रिंग स्टोरेज के लिए रेडिक्स पेड़।

181. मैट्रिक्स इनवर्सन और डिटरमिनेंट जैसे लिनियर अल्जेब्रा अवधारणाएं।

182. ग्राफ रंग और मैचिंग जैसे ग्राफ थ्योरी अवधारणाएं।

183. क्रिप्टोग्राफी और सुरक्षा में संख्या सिद्धांत के अनुप्रयोग।

184. रैंडमाइज्ड एल्गोरिथम और सिमुलेशन में प्रायिकता।

185. कंप्यूटर ग्राफिक्स और इमेज प्रोसेसिंग में ज्यामिति।

186. गणना और गणना समस्याओं में संयोजनिकी।

187. ऑपरेशन रिसर्च और लॉजिस्टिक्स में ऑप्टिमाइजेशन।

188. एल्गोरिथम विश्लेषण और डिजाइन के लिए डिस्क्रीट मैथ।

189. कुछ एल्गोरिथम में तेज गणनाओं के लिए बिटवाइज ऑपरेशंस का उपयोग करें।

190. स्टैक ओवरफ्लो से बचने के लिए मेमोरी उपयोग को ऑप्टिमाइज करें।

191. जब संभव हो, इनलाइन फंक्शंस और कंपाइलर ऑप्टिमाइजेशन का उपयोग करें।

192. अनावश्यक डेटा कॉपी से बचें और रेफरेंस या पॉइंटर्स का उपयोग करें।

193. बॉटलनेक और हॉटस्पॉट्स को पहचानने के लिए कोड प्रोफाइल करें।

194. मेमोइजेशन और कैशिंग का उपयोग करें ताकि परिणामों को स्टोर और पुनः उपयोग किया जा सके।

195. जब संभव हो, गति बढ़ाने के लिए गणनाओं को पैरललाइज करें।

196. जटिल समस्याओं को छोटे, प्रबंधनीय उपसमस्याओं में तोड़ें।

197. समस्या जटिलता को प्रबंधित करने के लिए अभिव्यक्ति का उपयोग करें।

198. एल्गोरिथमिक समाधानों को सरल बनाने के लिए गणितीय अवधारणाओं का उपयोग करें।

199. समस्या के दायरे को कम करने के लिए सममिति और अविचलन का उपयोग करें।

200. समस्या-समाधान कौशल को सुधारने के लिए लगातार अभ्यास और समीक्षा करें।