विभिन्न तरह की एल्गोरिदम को समझना

अनुष्का टोणपि

नवम्बर 2024 के अंक में प्रकाशित लेख [1] में हमने एल्गोरिदम की अवधारणा एवं इसके महत्त्व को जाना व समझा था। हमने देखा था कि अपने दैनिक जीवन में हम इसका किस तरह उपयोग करते हैं। इस लेख में हम इसके विभिन्न प्रकारों को जानेंगे व समझेंगे, और कक्षा में इन्हें प्रदर्शित करने लिए कुछ गतिविधियाँ सुझाएँगे।

एल्गोरिदम निर्देशों की एक सूची है जो किसी समस्या को हल करने या किसी कार्य को पूरा करने में मदद करती है। हमने पहले भी चर्चा की है कि किसी समस्या को हल करने के लिए कई एल्गोरिदम का प्रयोग किया जा सकता है। लेकिन क्या आपको पता था कि कई अलग-अलग प्रकार की एल्गोरिदम हैं जो किसी समस्या को हल करने के लिए विभिन्न तरीक़ों/विधियों का प्रयोग करती हैं? कुछ आपको चुनाव करने में मदद करती हैं, कुछ सबसे तेज़ रास्ता ढूँढ़ने में मदद करती हैं, और कुछ वस्तुओं को सही क्रम में रखती हैं। इस लेख में, हम मज़ेदार और इंटरैक्टिव तरीक़ों से सबसे आम प्रकार की एल्गोरिदम के बारे में जानेंगे।

चित्र-1 (Source: AI)

1. छँटाई एल्गोरिदम (Sorting algorithms) : चीज़ों को एक क्रम/श्रेणी में रखना

चलिए शुरुआत उसी से करते हैं जो हम सब करते हैं – व्यवस्थित करना! छँटाई एल्गोरिदम हमें वस्तुओं को एक ख़ास क्रम में व्यवस्थित करने में मदद करती है, जैसे कि सबसे छोटी से सबसे बड़ी, या वर्णानुक्रम अनुसार।

छँटाई करना क्यों महत्त्वपूर्ण है : जब आप अपनी किताबों को ऊँचाई के अनुसार, कपड़ों को रंग के अनुसार, या अपने क्रेयॉन को किसी इन्द्रधनुषी पैटर्न

चित्र-2

की-वर्ड : गणनात्मक सोच, प्रक्रियाएँ, खोजना, छँटाई, एल्गोरिदम

में व्यवस्थित कर रहे होते हैं, तो आप छँटाई कर रहे होते हैं। उदाहरण के लिए, कम्प्यूटर छँटाई का उपयोग ईमेल को तारीख़ के अनुसार, स्प्रेडशीट में संख्याओं, या खेलों में उच्च स्कोर के अनुसार व्यवस्थित करने के लिए करता है। **चित्र-2** में आप इन्द्रधनुष के रंगों के क्रम में व्यवस्थित छह क्रेयॉन देख सकते हैं।

वस्तुओं की छँटाई करने के कई अलग-अलग तरीक़े होते हैं। आगे छँटाई एल्गोरिदम की सूची दी गई है :

- **बुलबुले की तरह छँटाई (Bubble Sort) :** इस एल्गोरिदम में दो वस्तुओं के क्रमों/श्रेणी की आपस में तुलना की जाती है, और अगर वे ग़लत क्रम में होती हैं तो उनके क्रम/स्थान की अदला-बदली की जाती है। हर चरण में सबसे बड़ी वस्तु अन्त तक पहुँचा दी जाती है। उदाहरण के लिए, मान लीजिए हमारे पास बेतरतीब क्रम में कुछ संख्याएँ हैं : 1, 3, 4, 5, 2; और हम उन्हें आरोही क्रम (कम से ज़्यादा) में व्यवस्थित करना चाहते हैं। तब पहले चरण में हम पहली दो संख्याओं 1 और 3 पर विचार करते हैं। चूँकि वे क्रम में हैं, इसलिए हम उनकी जगह आपस में नहीं बदलेंगे। इसी तरह, हम अगली दो संख्याओं 3 और 4, या यहाँ तक कि 4 और 5 को भी आपस में नहीं बदलेंगे। लेकिन 5 और 2 क्रम में नहीं हैं, इसलिए हम 5 और 2 की जगह एक दूसरे से बदल देंगे। इस तरह हमें 1, 3, 4, 2, 5 मिलेंगे। अब तुलना की यही प्रक्रिया फिर दोहराएँगे। यानी एक बार फिर, शुरुआत से संख्या जोड़ों की आपस में तुलना करते जाएँगे। हम पाते हैं कि 4 और 2 क्रम में नहीं है तो उन्हें हम आपस में बदल देंगे। इस तरह हमें 1, 3, 2, 4, 5 मिलते हैं। अगले चरण की तुलना में, हम 3 और 2 को आपस में बदल देते हैं जिससे हमें 1, 2, 3, 4, 5 मिलते हैं। ध्यान दें कि कैसे संख्या 2 बाईं ओर खिसकती जाती है और 5 (सबसे बड़ी संख्या) अन्त में सरकती जाती है।
- **चुनना (Selection Sort) :** यह एल्गोरिदम पूरी सूची में से सबसे छोटी वस्तु को ढूँढ़कर उसे सबसे शुरू में रख देती है। फिर अगले चरण में अगली सबसे छोटी वस्तु को तलाशती है और उसे दूसरे क्रम पर रखती है, और इसी तरह यह पूरी छँटाई करती है। अब हम अपने पुराने उदाहरण को लेते हैं, लेकिन इस बार संख्याएँ अलग तरीक़े से गड्डमड्ड करते हैं : 5, 3, 4, 1, 2। सबसे छोटी वस्तु 1 है, इसलिए हम उसे बाईं ओर सबसे ऊपर रखते हैं : 1, 5, 3, 4, 2। फिर शेष वस्तुओं में से अगली सबसे छोटी वस्तु 2 है, जो 1 के बाद आती है : 1, 2, 5, 3, 4। इसी प्रकार, 3 और 4 भी अपने-अपने स्थानों पर तब तक रखते जाते हैं जब तक हमें पूरी तरह से क्रमबद्ध स्थिति 1, 2, 3, 4, 5 प्राप्त नहीं हो जाती।
- **प्रविष्ट (Insertion Sort) :** पहले से छँटी हुई सूची में प्रत्येक वस्तु को उसके सही स्थान पर प्रविष्ट करें। जैसे कि ताश खेलते समय हम अपने पत्तों को क्रम में जमाते हैं। मान लीजिए हमारे पास पहले वाले उदाहरणों में ली गई संख्याएँ ही हैं, लेकिन इस बार उनका क्रम है : 3, 1, 4, 2, 5।

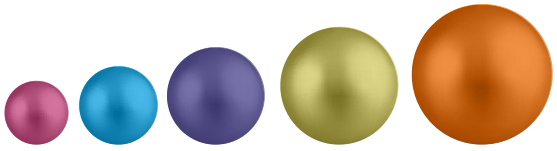
प्रविष्ट : मिश्रित ताश के पत्तों को क्रम में लगाना!

चरण-1	3				1	4	2	5
चरण-2	1	3			4	2	5	
चरण-3	1	3	4		2	5		
चरण-4	1	2	3	4	5			
चरण-5	1	2	3	4	5			

चित्र-3 : प्रविष्ट : यहाँ हरा रंग उन ताश के पत्तों को दर्शा रहा है जो हाथ में हैं, और गुलाबी उन पत्तों को जो बाएँ से दाएँ के क्रम में उठाए जा रहे हैं।

कक्षा में की जाने वाली गतिविधि : 1 से 10 तक के संख्या कार्ड लेकर उन्हें फेंट लें। अब हरेक छँटाई विधि को आज़माएँ। कौन-सी विधि कम चरणों में पूरी होती है? कौन-सी विधि ज़्यादा आसान है?

दी गई तीनों छँटाई एल्गोरिदम को अलग-अलग आकार की पाँच गेंदों के साथ आज़माएँ जिन्हें सबसे छोटी (अवरोही) से बड़ी के क्रम में व्यवस्थित करना है।



चित्र-4 : सबसे छोटी से सबसे बड़ी गेंदें जो छाँटी गई सूची में हैं।

कौन-सी एल्गोरिदम सबसे तेज़/सबसे आसान है?

2. खोजना एल्गोरिदम (Searching algorithms) : जो आपको चाहिए उसे ढूँढ़ना

कल्पना कीजिए कि आप अपने स्कूल बैग में अपनी गणित की नोटबुक ढूँढ़ रहे हैं। आप हर वस्तु को तब तक देखते हैं जब तक आपको वह मिल न जाए। यह एक रैखिक खोज है!

रैखिक खोज वह होती है जब किसी विशिष्ट वस्तु को खोजने के लिए कई प्रकार की वस्तुओं को ध्यान से छाँटा जाता है।

खोजना एल्गोरिदम के प्रकार :

रैखिक खोज (Linear Search) : प्रत्येक वस्तु को एक-एक करके देखना। यह धीमी है, लेकिन ग़ैर-क्रमबद्ध/ ग़ैर-श्रेणीबद्ध की गई सूचियों के लिए काम करती है।

द्विआधरी खोज (Binary Search) : यह सिर्फ़ क्रमबद्ध/ श्रेणीबद्ध की गई सूचियों के लिए काम करती है। मान लीजिए आप सबसे छोटी से सबसे बड़ी के क्रम में व्यवस्थित संख्याओं की सूची में से किसी एक ख़ास संख्या को ढूँढ़ रहे हैं। यदि आपकी संख्या छोटी है तो सूची के बाएँ ओर के आधे हिस्से में ढूँढ़ें; यदि वह बड़ी है तो सूची के दाएँ ओर के आधे हिस्से में ढूँढ़ें। इस प्रक्रिया को तब तक दोहराते रहें जब तक आपको वांछित संख्या न मिल जाए। ध्यान दें कि यह एल्गोरिदम केवल संख्याओं के लिए ही नहीं, और भी कई वस्तुओं के लिए काम करता है! मान लीजिए आप अपने अँग्रेज़ी शब्दकोश में 'Parrot' शब्द का अर्थ देखना चाहते हैं। चूँकि शब्दकोश बहुत मोटा है और वर्णमाला क्रम के अनुसार व्यवस्थित हज़ारों शब्दों से भरा है। आप जानते हैं कि शुरुआत से पन्ने पलट-पलटकर ढूँढ़ने में काफ़ी समय लगेगा। इसलिए आप चतुराई से काम लेते हैं। आप शब्दकोश को ठीक बीच से खोलते हैं और उस पन्ने पर 'Lion' शब्द पाते हैं। आप जानते हैं कि वर्णमाला के क्रम में 'Parrot' शब्द 'Lion' शब्द के बाद आता है, इसलिए आप शब्दकोश के पहले भाग को नज़रअन्दाज़ कर देते हैं और केवल दूसरे भाग में शब्द तलाशेंगे। अब आप इस नए खण्ड के बीच में (शब्दकोश के आधे के आधे भाग में) जाते हैं और वहाँ 'Tiger' शब्द पाते हैं। चूँकि 'Parrot', 'Tiger' से पहले आता है, इसलिए अब आप 'Tiger' के बाद के सभी पन्नों को नज़रअन्दाज़ कर देते हैं और केवल 'Lion' व 'Tiger' के बीच आने वाले पन्नों को देखते हैं। आप यही प्रक्रिया दोहराते रहते हैं, हर बार बचे हुए भाग के बीच में खोलकर देखते हैं कि कौन-सा आधा भाग रखना है, जब तक कि आपको अन्ततः 'Parrot' शब्द मिल नहीं जाता।

इसे आज़माएँ : 1 से 50 तक संख्या लिखे हुए पत्तों के एक बेतरतीब ढेर में 27 संख्या वाला एक पत्ता छिपाएँ। फिर उसी पत्ते को पत्तों के एक क्रमबद्ध ढेर में सही जगह पर रखें। दोनों तरीक़ों से उसे ढूँढ़ने का प्रयास करें। अपनी इस खोज को विस्तार से बताएँ। किस ढेर में ढूँढ़ना आसान था? कौन-सी खोज तेज़ थी?

3. लालची एल्गोरिदम (Greedy algorithms): सबसे अच्छे को तुरन्त चुनना

इस उम्मीद में, कि सभी अच्छे विकल्प मिलकर सर्वोत्तम समग्र समाधान प्रदान करेंगे, लालची एल्गोरिदम हर चरण में जो सर्वोत्तम विकल्प चुन सकती है उसे चुनती है।

उदाहरण : कल्पना कीजिए कि आप अलग-अलग आकार के पेपरमिंट्स इकट्ठा कर रहे हैं। आप साइज़ में सबसे बड़ा पेपरमिंट चुनना चाहते हैं, लेकिन केवल तीन ही चुन सकते हैं। एक लालची रणनीति जो सबसे बड़ा नज़र आ रहा है उसे अन्य सभी से तुलना किए बिना तुरन्त चुन लेगी।

चित्र-5 : पेपरमिंट्स

गतिविधि : मान लीजिए कि आपका जन्मदिन है और उपहार के तौर पर ख़रीदने के लिए आपको खिलौने चुनना है। आपके बक्से में सीमित जगह है – केवल तीन खिलौनों के लिए। सम्भावित 5 खिलौनों की एक सूची बनाइए। आपको हर एक खिलौने के साथ खेलने में कितना मज़ा आता है, इस आधार पर उन्हें 1 से 10 तक एक अंक दीजिए। उदाहरण के लिए, अगर आपको पहेलियों से खेलना पसन्द है तो पहेली वाले खिलौने को 10 में से 9 अंक दीजिए, और अगर आपको खिलौने वाली कारों से खेलना पसन्द नहीं है तो कार को 10 में से 3 अंक दीजिए। यदि आप सबसे ज़्यादा अंक पाने वाले खिलौनों को चुनने के लिए लालची एल्गोरिदम का इस्तेमाल करते हैं तो आपके पास तीनों पहेलियों वाले खेल हो सकते हैं। अब चुनाव का एक और प्रयास करें जिसमें किसी खिलौने का दोहराव न हो। आपको एक अलग तरह का लालची समाधान मिलेगा (क्योंकि पिछले समाधान में केवल अधिकतम आनन्द प्राप्त करने और इसे जितना हो सके उतना अधिक बनाने के लिए उच्चतम स्कोर वाले खिलौने शामिल करते जाना था)।

अपने जीवन की किसी अन्य स्थिति के बारे में सोचिए जहाँ आप किसी समस्या को हल करने के लिए लालची एल्गोरिदम का उपयोग कर सकें।

4. पुनरावर्ती एल्गोरिदम (Recursive algorithms) : छोटी-छोटी समस्याओं को हल करके समस्या का समाधान करना

पुनरावर्ती एल्गोरिदम किसी समस्या का समाधान उसी समस्या के छोटे-छोटे संस्करण को हल करके करती है। यह उसी प्रकार है जिस प्रकार आप अपने छोटे भाई-बहन से मदद माँगते हैं और वह उनसे भी छोटे भाई-बहन से, और यह सिलसिला चलता रहता है।

उदाहरण : रशियन मैट्रियोश्का गुड़िया वे गुड़ियाँ हैं जो एक के अन्दर एक रहती हैं। किसी समस्या को मैट्रियोश्का गुड़ियों के एक सेट के रूप में सोचिए, उस एक समस्या के अन्दर छोटी-छोटी समस्याएँ बसी हुई हैं।

चित्र-6 : मैट्रियोश्का गुड़ियाँ
(source: Macalester College, Russian studies)

गतिविधि : जिस तरह नीचे दिखाया गया है उस तरह कप की एक मीनार बनाइए।

चित्र-7 : कप की एक मीनार

अब कपों को हटाने का कोई तरीक़ा सोचने की कोशिश कीजिए। सबसे नीचे बाईं ओर वाले कप को हटाने के लिए, आपको उसके ऊपर वाले कपों को हटाना होगा, और उन कपों को हटाने के लिए, आपको सबसे ऊपरी कप को हटाना होगा। इस तरह, कपों को हटाने की समस्या एक पुनरावर्ती एल्गोरिदम का उपयोग करके एक सरल समस्या में बदल जाती है, और जब आप समस्या का सबसे छोटा भाग हल कर लेते हैं, बाक़ी भाग क्रमिक रूप से हल होते जाते हैं। यानी जब आप ऊपरी कप को हटाते हैं, आप कपों की अगली पंक्ति हटाने के लिए आगे बढ़ सकते हैं, और अन्ततः कपों की आख़िरी पंक्ति हटा सकते हैं।

पुनरावर्ती एल्गोरिदम का उपयोग करके हल की जाने वाली समस्याएँ बड़ी कक्षाओं के लिए अधिक उपयुक्त हैं, इसलिए यहाँ उनके उदाहरण नहीं दिए जा रहे हैं।

5. हर सम्भव प्रयास करें (Brute force) : हर सम्भावना को आज़माना

हर सम्भव प्रयास का मतलब है हर सम्भावना को करके देखना। यह एक चतुर तरीक़ा तो नहीं है, लेकिन समाधान ज़रूर सुनिश्चित करता है।

उदाहरण : मान लीजिए किसी दरवाज़े पर एक ताला लगा हुआ है, और आपके पास चाबियों का एक सेट है जिसमें कुल 10 चाबियाँ हैं। आपको पता नहीं है कि किस चाबी से दरवाज़ा खुलेगा, इसलिए आपको हर एक चाबी को आज़माना होगा, और उस ताले में हर चाबी लगाकर पता लगाना होगा कि किससे दरवाज़ा खुलेगा।

गतिविधि : विद्यार्थियों को 1 से 20 तक की संख्या में से कोई एक संख्या सोचने दीजिए और उनके साथी को हर सम्भव प्रयास विधि का उपयोग करके सोची गई संख्या का अनुमान लगाने दीजिए।

सही संख्या पता करने में कितने अनुमान लगे?

निष्कर्ष : एक एल्गोरिदम की तरह सोचें!

एल्गोरिदम हर जगह मौजूद हैं। वे हमें अधिक व्यवस्थित, अधिक कुशल, और अधिक रचनात्मक बनने में मदद कर सकती हैं। विभिन्न प्रकार की एल्गोरिदम सीखकर आप एक बेहतर विचारक और समस्या समाधानकर्ता बन सकते हैं, क्योंकि आप इन एल्गोरिदम की मदद से किसी समस्या को हल करने के लिए विभिन्न तरीक़ों का उपयोग कर सकते हैं।

अगली बार जब आप किसी मुश्किल चुनौती का सामना करें तो ख़ुद से पूछें : क्या मैं इसे चरणों में तोड़ सकता हूँ? क्या मैं लालची, हर सम्भव प्रयास या पुनरावर्ती एल्गोरिदम जैसी विभिन्न विधियाँ आज़मा सकता हूँ?

क्योंकि जब आप एक एल्गोरिदम की तरह सोचते हैं, कोई भी समस्या बहुत बड़ी नहीं लगती!

सम्पादक की टीप : किसी के मन में यह ख़्याल आ सकता है कि क्या इन एल्गोरिदम को प्राथमिक कक्षाओं में ही पढ़ाना शुरू कर दिया जाना चाहिए? हम निश्चित रूप से तलाश करने के लिए एल्गोरिदम का नाम बताने और परिभाषा देने की सिफ़ारिश नहीं करेंगे; हालाँकि, आपने देखा होगा कि उपयोग किए गए अधिकांश उदाहरण प्राथमिक विद्यालय के बच्चों के दैनिक जीवन से हैं। ऐसे कार्यों की रूपरेखा तैयार करना दिलचस्प होगा जिनमें खोजने की आवश्यकता होती है – यदि विद्यार्थी इन विभिन्न विधियों का उपयोग कर सकें, और कार्य के बाद चिन्तन व चर्चा के दौरान उनकी तुलना कर सकें तो वे कम्प्यूटेशनल सोच के विचारों को ग्रहण कर सकेंगे। वे एक मज़ेदार तरीक़े से और बिना किसी डर के इसकी प्रासंगिकता को समझ सकेंगे।

Reference

1. Tonapi, A. (2024). Introduction to algorithms. *At Right Angles, (20),* 14–19. https://bit.ly/3XJ6E1e

अनुष्का टोणपि बेंगलूरु के अल्पाइन पब्लिक स्कूल में ग्यारहवीं कक्षा की छात्रा हैं। वे न्यूयॉर्क एकेडमी ऑफ़ साइंसेज की एक युवा सदस्य हैं और स्पिरिट ऑफ़ रामानुजन फ़ेलोशिप पुरस्कार विजेता हैं। उन्हें गणित के प्रश्न हल करने में मज़ा आता है। थ्योरिटिकल कम्प्यूटर साइंस में उनकी रुचि है। अनुष्का को शतरंज खेलना बहुत पसन्द है। वे अपने ख़ाली समय में कर्नाटक संगीत का अभ्यास करती हैं। उनसे Anushka.tonapi@gmail.com पर सम्पर्क किया जा सकता है।

अनुवाद : प्रियेश गुप्ता **पुनरीक्षण :** प्रतिका गुप्ता **कॉपी एडिटर :** अतुल अग्रवाल