**UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”**

**FAKULTETI I SHKENCAVE MATEMATIKE – NATYRORE**

DEPARTAMENTI I MATEMATIKËS

Programi: Shkenca kompjuterike



Projekt

***LËNDA:*** *Inteligjenca Artificiale*

***TEMA:*** *Implementimi i problemit të numrave të Langford-it në gjuhën programuese Java*

Profesori: Studentet:

Arbana Grabanica

Besnik Duriqi Suzana Ternava

Prishtinë, 2023

# Abstrakt

Ky punim paraqet një analizë të problemit apo sekuencës së Langford-it si dhe zgjidhjes së tij duke përdorur gjuhën programuese Java. Problemi i Langfordit është një problem i cili përfshinë krijimin e një vargu prej n numrave ( gjatësia e vargut është 2\*n) ku secili numër gjendet 2 herë në atë varg teksa distanca mes numrave të njëjtë është vet ai numër. Në fillim të këtij projekti jepet një histori e problemit të Langfordit si dhe një kronologji e përparimit në zgjidhjen e këtij problemi përgjat viteve. Pas kësaj tregohet edhe për aplikimin e këtyre vargjeve në problemet reale. Për implementimin e këtij problemi në gjuhën Java kemi përdorur algoritmin Hill Climbing ku sqarohet saktësisht se si vepron ky algoritëm. Në këtë punim krahasohen edhe algoritmi Hill Climbing dhe një algoritëm tjetër specifik i dhënë. Në fund jepen rezultatet e fituara pas ekzekutimit të programit në Java varësisht nga të hyrat e algoritmit.

# Përmbajtja

[Abstrakt 2](#_Toc104729286)

[Përmbajtja 3](#_Toc104729287)

[Hyrje 4](#_Toc104729288)

[Problemi i Langford-it 6](#_Toc104729289)

[Përdorimi i numrave të Langford-it 7](#_Toc104729290)

[Qasja dhe metodologjia e projektit 8](#_Toc104729291)

[Hill Climbing vs Brute Force 10](#_Toc104729292)

[Rezultatet 11](#_Toc104729293)

[Përfundimi 11](#_Toc104729294)

[Referencat 12](#_Toc104729295)

# Hyrje

Problemi i Langfordit ka marrë emrin e matematikanit skocez Dudley Langford, i cili dikur vëzhgoi djalin e tij duke luajtur me disa blloqe me ngjyra. Ai vuri re se fëmija i kishte rregulluar katër palë blloqe me ngjyra të kuqe, të kalter, te zezë dhe të verdhë në mënyrë që të kishte një bllok midis çiftit të zi, dy midis çiftit të kuq, tre midis çiftit të kaltër dhe katër midis çiftit të verdhë [1]. Për ta kuptuar më mirë këtë shih figurën 1.

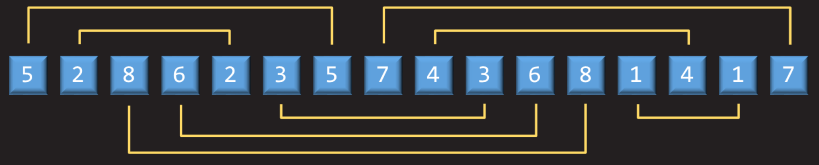


**Figura 1:** Renditja e blloqeve sipas Langford-it [1].

Pas kësaj, Langford e gjeneralizoj këtë problem nga ngjyrat në numra. Duke u bazuar në figurën 1, ai krijoi vargun 4-1-3-1-2-4-3-2 dhe u simbolizua me shënimin L(k,n) ku parametri i parë, k, tregon numrin e përsëritjeve të secilit numër dhe parametri i dytë, n, tregon se sa numra janë në varg. Kjo do të thotë se gjatësia e vargut gjithmonë është . Në rastin tonë k = 2 dhe n = 4, pra L(2,4).

Një kronologji e avancimit të zgjidhjeve të vargjeve të Langfordit përgjat viteve është paraqitur në Tabelën 2.

Një lloj i veçantë i paraqitjes së vargjeve të tilla është zgjidhja Planare. Ky lloj i paraqitjes mundëson që të çiftet të lidhen nga linjat duke mos u kryqëzuar fare, siç shihet në figurën 2.



**Figura 2 :** Paraqitja Planare e vargjeve të Langford-it për n = 8 [2].

Numri i zgjidhjeve Planare për vargjet e Langford-it është shumë i vogël në krahasim me numrin total të zgjidhjeve. Për shembull, për n=11, vetëm 16 nga 17792 zgjidhje totale janë Planare. Për n=3 kemi vetëm një zgjidhje e cila njëkohësisht është Planare [4].

Për të parë numrin e zgjidhjeve Planare për vlera të N = 3, 4, 7, 8, 11, 12, 15, 16, 19, 20, 23, 24, 27, 28, 31 shih Tabelën 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N =** | **3** | **4** | **7** | **8** | **11** | **12** | **15** | **16** | **19** | **20** | **23** | **24** | **27** | **28** | **31** |
| Planare | 1 | 0 | 0 | 4 | 16 | 40 | 194 | 274 | 2384 | 4719 | 31856 | 62124 | 426502 | 817717 | 5724640 |

**Tabela 1:** Zgjidhjet Planare për L(2,N).

|  |  |
| --- | --- |
| Viti - Llogaritja | Përshkrimi |
| 1999 - L(2, 19) | Ekipi Groth ( Bob Dumas, Edward Groth, Nevin Savage dhe Rick Groth) zgjidhën L(2,19) duke përdorur disa kompjutera. |
| 2002 - L(2, 20) | Mike Godfrey përcaktoi L(2,20) me metodën e re algjebrike e cila ishte shumë më e shpejtë se algoritmi klasik i kërkimit. |
| 2004 – L(4, 24) | Richard Noble numëroi të tri zgjidhjet L(4, 24) për afër dy vitesh kohë. |
| 2004 – L(2, 23) | Një eksperiment nga 5 persona në Universitetin e Reims përdori metodën e Godfrey për të llogaritur L(2, 23) duke përdorur një përzierje të 30 makinave Intel dhe Sun. |
| 2005 – L(2, 24) | Michael Krajecki, Christophe Jaillet, Alain Bui llogaritën L(2, 24) për 3 muaj duke përdorur 12 – 15 procesora. |
| 2006 – 2014 | Asgjë e re e dokumentuar. |
| 2015 – L(2, 27), L(2, 28) | Ekipi Krajecki llogariti L(2, 27).  Ekipi Assarpour-Liu llogariti L(2, 28). |
| 2017 – Zgjidhjet Planare ( P(2, 28)) | Boris Dimitrov paraqiti zgjidhjet Planare P(2, 28). |
| 2018 – P(2,31-32) | Rory Molinari llogariti P(2, 31) dhe P(2, 32). |
| 2019 – Varianti Kolumbian | Varianti Kolumbian paraqet zgjidhjet e Langford-it në atë mënyrë që asnjë çift i plotë më i vogël se n-1 nuk është brenda n-1. Ky variant u përgatit nga Bernardo Santtos dhe Freddy Barrera nga Kolumbia. |
| 2021 – Formula asimptotike që vlerëson L(2, n). | Në vitin 2021 u nxorr një formulë për llogaritjen e numrit të vargjeve të Langford-it. |

**Tabela 2:** Historia e zgjidhjeve të vargjeve të Langford-it përgjat viteve [3].

# Problemi i Langford-it

Nga hyrja e këtij punimi, pamë se kemi të bëjmë me një problem L(k,n) ku duhet të rregullojmë k bashkësi të numrave 1 deri në n ashtu që çdo numër m është m numra larg nga çifti i saj. Vargjet e para që u krijuan ishin për n = 3 dhe n = 4. Langford i vazhdoi këto vargje të tilla të numrave edhe për gjatësitë 7,8,11,12 dhe 15. Mirëpo ai pa se këto sekuenca nuk ishin të mundshme për numrat 5,6,9, apo 10. Prandaj ai parashtroi një pyetje se për cilat vlera të n, mund të zgjidhim vargje të tilla [6]?

Zgjidhjen e këtij problemi e bëri Roy O.Davies i cili vërtetoi se numri n duhet të jetë i plotëpjestueshëm me 4. Pra, që të krijojmë një varg të Langford-it duhet që numrin e elementeve në varg ta zgjedhim të trajtës ose [5].

Ai erdhi në këtë përfundim kështu:

Së pari, nëse kemi një varg të Langford-it për n numra, atëherë vargu do të ketë gjatësinë .

Në qoftë se, shënojmë me Ak pozitën e ndonjë numri k, atëherë në pozitën Bk do të jetë çifti i k-së. Pra, 1.

Pozita: 1 2 3 . . . Ak  Ak+1 . . . Bk 2\*n

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | … | k |  | … |  | K |  |  |

Numrat:

**k – njësi**

Dijmë se

Nëse zëvendësojmë shumën (2) në ekuacionin (1), fitojmë:

Atëherë, kemi:

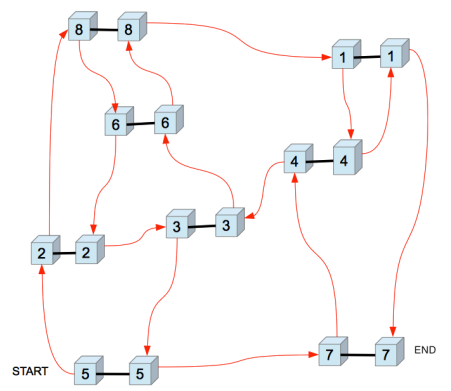
Meqë duhet të jetë numër i natyror, sepse pozitat janë numra natyror, atëherë n\*(3n-1) duhet të jetë i plotëplestueshëm me 4. Nga kjo, përfundojmë se n = ose n= – 1, [2].

## Përdorimi i numrave të Langford-it

Sa i përket përdorimit të numrave të Langford-it, përveç në sferën e matematikës, konkretisht kombinatorikës sepse ka të bëj me permutacione të numrave, këto vargje janë përdorur edhe për disa probleme të tjera. Siç shihet në figurën 4, ku është paraqitur e ashtuquajtura Ura e Konigsbergut, numrat e Langfordit janë përdorur për të kaluar nëpër të gjitha qytet por pa ecur mbi asnjë urë. Ky problem ishte një problem shekullor dhe një zgjidhje për këtë problem, ishin numrat e Langfordit. Kështu, ata meqë kishin 16 vende për të kaluar pa kaluar nëpër ura, atëherë ata zgjidhen numrat e Langfordit për 8 çifte, ku secili çift bashkohej me nga një urë. Ata zgjodhën vargun 5-2-8-6-2-3-5-7-4-3-6-8-1-4-1-7 si zgjidhje për këtë problem. Një tjetër aplikim i numrave të Langfordit është edhe në logon e website-it për arkivat e historisë së matematikës MacTutor i cili zgjodhi inversin e vargut 4-1-3-1-2-4-3-2 (shih figuën 3) [4].



**Figura 3:** Përdorimi i vargut Langford në logon e MacTutor [4].



**Figura 4:** Zgjidhja e problemit “Ura e Konigsbergut” me numrat e Langford-it [4].

# Qasja dhe metodologjia e projektit

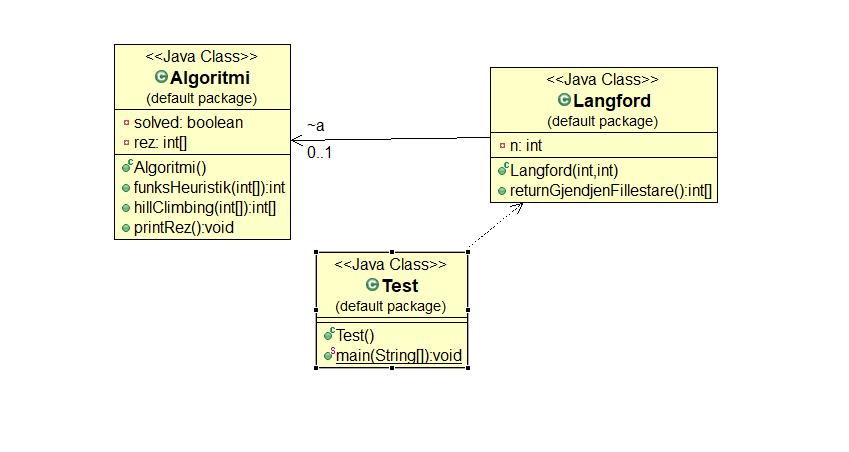
Deri tani, u familjarizuam me problemin e Langfordit, duke përdorur edhe shembuj të aplikimit në problemet e jetës së përditshme. Tani, duhet që këtë problem ta zgjidhim në gjuhën programuese Java përmes një algoritmi e në këtë rast Algoritmin Hill Climbing.

Algoritmi Hill Climbing është një algoritëm kërkimi lokal, i cili vazhdimisht lëviz në drejtim të rritjes së lartësisë/vlerës për të gjetur “majën e malit” ose zgjidhjen më të mirë të problemit. Algoritmi përfundon kur arrin në një vlerë kulmore ku asnjë fqinj nuk ka vlerë më të lartë [8].

Ky algoritëm zakonisht përdoret për optimizimin e problemeve matematikore [7].

Në lidhje me problemin tonë, si input algoritmi merr një varg të numrave të plotë, nëse vargu është zgjidhja e kërkuar e kthen vargun si zgjidhje, përndryshe rastësisht zgjedh një element të vargut dhe e gjen elementin tjetër të barabartë me të dhe nëse nuk është në pozitën përkatëse elementi ndërron pozitën me elementin i cili gjendet në pozitën përkatëse. Algoritmi pastaj e krahason vargun aktual me vargun paraprak, nëse vargu aktual ka funksion heuristik më të madh se vargu paraprak algoritmi fillon përsëri me vargun aktual si input. Algoritmi vazhdon të përsëritet derisa të gjendet zgjidhja e kërkuar ose derisa algoritmi të ngec në një maksimum lokal.

Diagrami i klasave të përdorura për implementim, është paraqitur në figurën 5.



**Figura 5:** Diagrami i klasave të përdorura

Nëse algoritmi nuk gjen zgjidhjen e problemit e ristartojmë nga fillimi derisa të gjejë zgjidhjen ose deri sa ta arrin numrin maksimal të ristartimeve, nëse pas arritjes së numrit maksimal të ristartimeve algoritmi nuk arrin ta gjejë zgjidhjen e kërkuar si output marrim tekstin “Algoritmi dështoi.” Dhe vargun e fituar në provën e fundit të algoritmit për të gjetur zgjidhjen.

Pseudokodi për zgjidhjen e problemit është ky:

|  |
| --- |
| Algoritmi 1: Pseudokodi per zgjidhjen e problemit sipas algoritmit Hill Climbing |
| procedure HillClimbing (GjendjaFillestare)  GjendjaAktuale = GjendjaFillestare  if funksHeuristik(GjendjaAktuale) = GjendjaAktuale.length/2  then  return GjendjaAktuale  end if  i = Random().nextInt(GjendjaAktuale.length)  if i+GjendjaAktuale[i]+1 < GjendjaAktuale.length AND  GjendjaAktuale[i] ≠ GjendjaAktuale [i+1+GjendjaAktuale[i]]  then  index = 0  for j= 0 to GjendjaAktuale.length do  if GjendjaAktuale[j] = GjendjaAktuale[i] AND j ≠ i  then  index = j  end for  end if  end for  GjendjaAktuale[i+1+GjendjaAktuale[i]] ↔ GjendjaAktuale[index]  end if  else if i-(GjendjaAktuale[i]+1) ≥ 0 AND  GjendjaAktuale[i] ≠ GjendjaAktuale[i-(GjendjaAktuale[i]+1)]  then  index = 0  for j= 0 to GjendjaAktuale.length do  if GjendjaAktuale[j] = GjendjaAktuale[i] AND j ≠ i  then  index = j  end for  end if  end for  GjendjaAktuale[i+1+GjendjaAktuale[i]] ↔ GjendjaAktuale[index]  end if  if funksioniHeuristik(GjendjaFillestare) < funksioniHeuristik(GjendjaAktuale)  then  HillClimbing(GjendjaAktuale)   |  | | --- | | end procedure | |

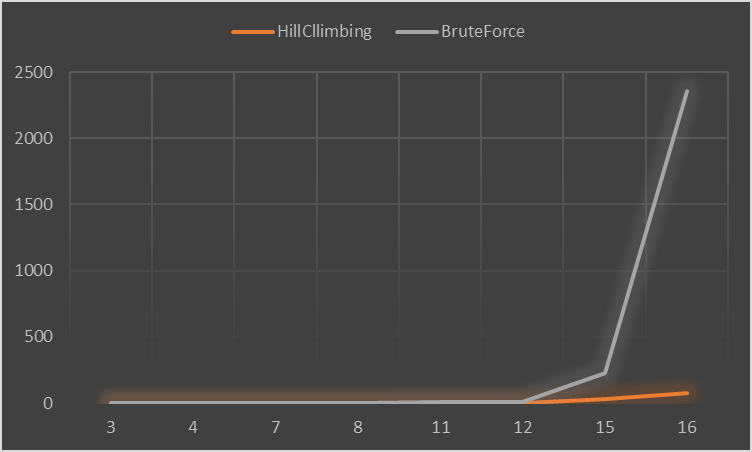
## Hill Climbing vs Brute Force

Një tjetër mënyrë e llogaritjes së numrave të Langford-it, është përmes metodës së Brute Force. Kjo metodë llogarit të gjitha rastet e mundshme derisa gjen zgjidhjen e duhur. Për këtë shkak, Brute Force merr kohë më shumë për llogaritje dhe kësisoj është algoritëm apo metodë “më e keqe” sesa Hill Climbing. Një krahasim sipas kohës është dhënë në Tabelën 3. Shihet se përderisa vlera n është më e vogël se 15, dallimi nuk është aq i madh. Dallimi vërehet pas vlerës 15, ndërsa kur n=24 atëherë Hill Climbing nuk shfaq statistika të mira. Për ta parë edhe më mirë dallimin mes Hill Climbing dhe Brute Forces atëherë shikoni grafikun në figurën 6.

Nga kjo figurë shohim se sa kohë e madhe i duhet Brute Forces të gjej zgjidhje për n > 15 për dallim nga Hill Climbing.

|  |  |  |
| --- | --- | --- |
| Vlera N | Koha e zgjidhjes për Hill Climbing | Koha e zgjidhjes për Brute Force |
| 3 | 0.001163 ms | 0.2 ms |
| 4 | 0.008635 ms | 0.21 ms |
| 7 | 0.11636 ms | 0.43 ms |
| 8 | 0.13694 ms | 0.56 ms |
| 11 | 1.397 ms | 9.43 ms |
| 12 | 3.0597 ms | 4.28 ms |
| 15 | 33.177 ms | 190.913 ms |
| 16 | 77.0176 ms | 2279.611 ms |
| 19 | 861.952 ms | 63480.6 ms |
| 20 | 2285.18 ms | 620274 ms |
| 23 | 33974.75 ms | 83084.5 ms |
| 24 | 73423.85417 ms | 88620.3 ms |

**Tabela 3:** Hill Climbing vs Brute Force



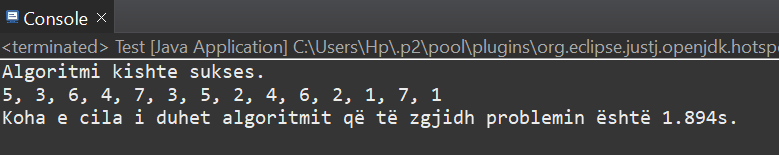
***Milisekondat***

***Vlerat n***

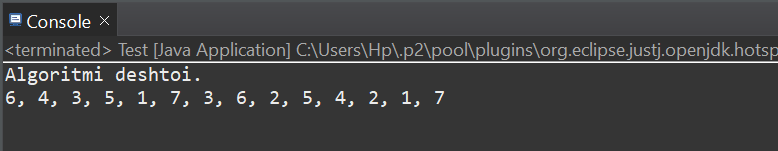
**Figura 6:** Hill Climbing vs Brute Force

# Rezultatet

Në këtë kapitull, do të shfaqim rezultatet e fituara nga ekzekutimi i programit në Java. Pasi e ekzekutojmë kodin me input maxRestarts si numër natyrorë dhe n, ku n = 4k ose n = 4k-1, nëse ka zgjidhje për numrin e caktuar të provave në console na shfaqet output zgjidhja dhe koha e cila i duhet algoritmit për ta zgjidhur problemin ngjashëm si në figurën 7. Por nëse algoritmi nuk mund ta zgjidh problemin per numrin e caktuar të përsëritjeve në console na shfaqet mesazhi që algoritmi dështoi si dhe rezultati i provës së fundit si në figurën 8. Nëse si input japim maxRestarts = -1 dhe n, ku n = 4k ose n = 4k-1 algoritmi përsëritet deri sa të gjendet zgjidhja dhe në console shfaqet outputi i ngjashëm si në figurën 7.

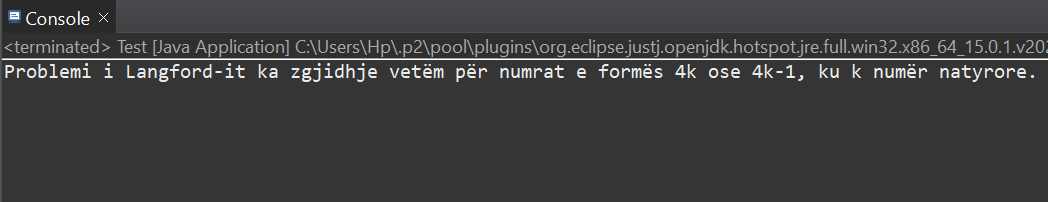


**Figura 7:** Gjetja e vargut të Langfordit kur n=7.



**Figura 8:** Afishimi pas dështimit të algoritmit.

Nëse e ekzekutojmë kodin me input n, ku n != 4k ose n != 4k-1 në console na shfaqet output që problemi nuk ka zgjidhje për n-in e dhënë dhe na tregon për qfarë forma të n-it ka problemi zgjidhje në figurën 9.



**Figura 9:** Mesazhi pas gabimit në input.

# Përfundimi

Sa i përket këtij projekti, qëllimi ynë ka qenë krijimi i një programi përmes Java-s për zgjidhjen apo gjenerimin e vargjeve të Langfordit, vargje këto specifike siç u trajtuan edhe më lart në pjesën e hyrjes dhe shtjellimit të problemit të Langfordit. Pamë se përdorëm algoritmin Hill Climbing, algoritëm ky i cili në programin tonë merrte si input një vlerë të numrit të vargut si dhe numrin maksimal të përsëritjeve apo ristartimeve të algoritmit. Krahasuam algoritmin Hill Climbing dhe Brute Force, ky i fundit që përdorte të gjitha rastet e mundshme dhe si përfundim arritëm që kur parametri n merrte vlera më të mëdha se 16 atëherë intervali kohor në milisekonda i gjetjes së zgjidhjes nga Brute Force ishte shumë i madh në krahasim me Hill Climbing. Krejt në fund, pas testimeve të bëra, arritëm në përfundimin se algoritmi Hill Climbing nuk është algoritëm aq efikas për zgjidhjen e problemeve komplekse sepse kur parametri n është i madh i duhet shumë kohë për të gjet zgjidhje.

# Referencat

[1] Timehaven J. (2019, March 5). *Langford's Problem*. Retrieved from What is Langford's Problem!?: http://www.dialectrix.com/langford.html

[2] DataGenetics. (2020, May 9). *Langford's Sequences*. Retrieved from https://datagenetics.com/blog/october32014/index.html

[3] Langford Timeline of Activity. (2021, July 11). Retrieved from http://www.dialectrix.com/langford/Timeline.html

[4] Miller J. (2020, October 16). *Langford's Problem, Remixed*. Retrieved from https://www.gathering4gardner.org/g4g11gift/Miller\_John-Langfords\_Problem Remixed.pdf

[5] Ben - Ari M. . (2021, November 8). *Langford’s Problem*. Retrieved from https://www.weizmann.ac.il/sci-tea/benari/sites/sci-tea.benari/files/uploads/softwareAndLearningMaterials/langford-en.pdf

[6] Walsh T. . (2020, February 9). *Langford's number problem*. Retrieved from https://www.csplib.org/Problems/prob024/

[7] Jaillet Ch. & Krajecki M. (2018, June 25). *Solving the Langford problem in parallel*. Retrieved from http://www.dialectrix.com/langford/krajecki/jaillet\_krajecki.pdf

[8] Javatpoint. (2021, December 23). *Hill Climbing Algorithm in Artificial Intelligence*. Retrieved from https://www.javatpoint.com/hill-climbing-algorithm-in-ai